KERNFORS

Development and Improvement of Identification Methods for Time Varying and Nonlinear Industrial Processes

Darmstadt
March 1989

IN SCIENTIFIC RESEAR

GERMAN-YUGOSLAV COOPERATION
IN SCIENTIFIC RESEARCH AND TECHNOLOGICAL DEVELOPMENT

1988 REPORT ON PROJECT

DEVELOPMENT AND IMPROVEMENT OF IDENTIFICATION METHODS FOR TIME VARYING AND NONLINEAR INDUSTRIAL PROCESSES

PROJECT NR. 32.1.I5A.6A

Bilateral Cooperation

TECHNISCHE HOCHSCHULE DARMSTADT
INSTITUT FÜR REGELUNGSTECHNIK
FACHGEBIET REGELSYSTEMTECHNIK

and

UNIVERZA "EDVARDA KARDELJA" LJUBLJANA
INSTITUT JOŽEF STEFAN
ODSEK ZA RAČUNALNIŠKO AVTOMATIZACIJO IN REGULACIJE

supported by

INTERNATIONAL BUREAU, KFA JÜLICH

DARMSTADT, MARCH 1989

Herausgegeben von der Kernforschungsanlage Jülich GmbH
ZENTRALBIBLIOTHEK
Postfach 1913 · D-5170 Jülich
Telefon 02461/61-0 · Telex 833556-70 kfa d

Titelsatz: Graphische Betriebe der KFA

Druck: WEKA-Druck GmbH, Linnich

Jül-Spez-507
ISSN 0343-7639
ISBN 3-89336-022-0

CONTENTS

SOME NOTES ON IMPLEMENTATION OF THE CACSD PACKAGE ANA ON IBM-PC COMPUTERS AND THE CONCEPT OF GENERAL DYNAMIC STRUCTURE SIMULATION

1. INTRODUCTION

In the frame of the project "Identification of nonlinear and time-variable processes" up to now mainly investigations concerning study and development of some identification methods have been performed together with evaluation of these methods on simulated as also on real processes. To solve these problems a great number of different computer programs were developed which, in our opinion, should be connected into an independent program package or incorporated into one of the existing one. In the same time on the Yugoslav side the general purpose CACSD package, called ANA was developed. This package appears to be a sensible frame for inclusion of developed computer programs, as well as a good basis for a common development of a new and advanced software tool. The last, as we agreed with our German colleges, should be a theme of further cooperation between Technische Hochschule Darmstadt and Josef Stefan Institute in Ljubljana.

Unfortunately, both sides developed the software on the different computers and under different operating systems. Therefore a kind of unification has become an imperative. In this sense as a good solution IBM-PC and compatible computers, running under MS-DOS operating system, were chosen as a common basis for further development of software, due to inexpensiveness and good performances.

Following this idea, the first step was the implementation of the existing package ANA, which was developed on DEC computer under operation system RT-11, on the IBM-PC computer.

In the first part of this report some problems of the transfer of the package ANA to the IBM-PC computer and special features considering this installation are commented.

The second part of the report includes a presentation of a special function in ANA, namely, the concept of the general dynamic simulation

structure which is, in its present state, a good frame for solving identification problems of nonlinear and time varying processes.

Finally, in the appendix a paper presented at the 12-th World Congress on Scientific Computation, IMACS - 88 in Paris is given. It illustrates some features and capabilities of the program package ANA.

2. IMPLEMENTATION OF THE PROGRAM PACKAGE ANA ON THE IBM-PC COMPUTERS

The interactive program package for analysis and design of control systems ANA has been up to now implemented under four different operation systems (RT-11, RSX-11, VMS and DELTA M). With the development of personal computers (especially IBM-PC) and their wide use in almost all areas of human work, the implementation of the program package ANA on MS-DOS has become very interesting.

On the basis of an analysis it has been found out that the standard hardware and software of the IBM-PC computers enables realization of all existing functions of the program package ANA. However, these are certain advantages which must be taken into account. The main advantage is the relatively inexpensive and standard graphical support on this computers. So, with installation of a proper software we can obtain an effective, low-cost, CACSD work station.

In the following paragraphs some characteristics of the IBM-PC installation are given and some instruction which users should follow to use the package successfully.

2.1. Required hardware and software

The following hardware and software is required for the installation of the program package ANA on the personal computer IBM PC (and compatibles):

a) hardware

- personal computer IBM PC - XT or IBM PC -AT,
- Hard disc with the capacity of more than 6.5 Mbytes,
- RAM memory more than 420 Kbytes,

- math. coprocessor (not obligatory but very recommendable),
- one of graphic interfaces (and corresponding monitor):
  Monochrome Graphic Adapter (HGC) with resolution 720x350,
  Colorgraphic Adapter (CGA) with resolution 640x200 (two colors),
  Enhanced graphic adapter (EGA) with resolution 640x350 (16 colors) - most recommendable,
- dot matrix printer (e.g. IBM graphics printer; EPSON printer family - FX80, FX80+, FX100, RX80, RX100, RX80F/T+, LX86, LX80, LQ1500; NEC printers - NEC P6, P7, STAR printer - NB - 15, SG - 10/15) (not obligatory but recommendable),
- plotter based on standard HP - GL language (e.g. Hewlett Packard - HP7475A, HP7585B, HP7220C, HP7045B) (not obligatory but recommendable).

b) software

- operating system DOS 3.1 (or later versions),
- FORTRAN compiler and linker with corresponding library (Microsoft (R) FORTRAN 77 V 3.31 - compiler; Microsoft (R) 8086 object linker V 3.04 - linker) (They are used for the operation SSI in the synthesis and are not needed if this operation is not included the package).

2.2. Package installation

The installation of the program package ANA can be performed only by the authorized persons (contractors). Certain specificities and customer wishes can be taken into account during the installation. After the installation the customer can not influence the options which are in the contractor domain. This is the reason for the customer to be cautious when performing some reorganizations of the computer the package is installed on. The correct operation of the package can not be reestablished by the user himself if any wrong operation was caused by any reason.

2.3. Package configuration for the concrete use

Before any use of the package the customer can adapt some working modes of the package to his wishes and hardware respectively. The possibilities which can be influenced by the user are conected with:

- mass storage (determination of the working and auxiliary directory),

- keyboard (setting of particular keys),
- terminal (determination of the indicators for data entering and colors setting for graphic mode plotting),
- plotter (determination of the output channel the plotter is connected to and pens to be used for the particular variables documentations)
- operation SSI in the synthesis branch (adjustment of the command procedures for compilation and linking).

All described settings are performed by the supplier and have to be changed only in the case of special wishes of the customer or in the case of computer structure changes.

Package configuration changes can be done by the customer by editing the corresponding files:

ANAASG.DAT- definition of the working and auxiliary directories,
TASTER.DAT- setting of the keys and indicators at the data entering,
GRAPH.DAT - setting of colors to be used for plotting on the screen,
PLOT.DAT - determination of the channel to be used for the plotter and the pens to be used for the documentation,
FORT.BAT and LINK.BAT -adjusting of command procedures for compiling and linking.

2.3.1. File ANAASG.DAT

File ANAASG.DAT must be on the directory the package is started from. It contains three records, which determine the ANA system directory, the working directory and the auxiliary directory in the form which is requested according to the operating system. The first record contains the ANA system directory name, the second the working directory name and the third the auxiliary directory name. The records must not exceed 16 characters. An example of the file image is in Fig. 1.

```
C:\ANA\EXE\
C:\WORK\
C:\WORK\BACK\
```

Fig.1: Print of the file ANAASG.DAT

In the example (Fig. 1) program package ANA is installed on the subdirectory C:\ANA\EXE\, the working directory is the subdirectory C:\WORK\ (it

is not necessary that it is the same as the subdirectory the package is started from) and the auxiliary directory is the subdirectory C:\WORK\BACK\.

If a new directory is created the existing version can be copied to the working directory and the second and the third record must be updated (the first one must remain unchanged). In the case of the auxiliary directory redefinition only the third record has to be updated.

WARNING:
ANA system directory must never be defined as the working directory, it is recommended also not to use ANA system directory as the auxiliary directory!

2.3.2. File TASTER.DAT

File TASTER.DAT is on ANA system directory and contains all necessary parameters for correct operation of the package during interactive entering of the data (cf. 3.3 in ANA User Manual).

Content of the file consists of two groups of records:

a) First four records must always exist and determine the following four indicators in turn:
   - uncurrent unselected data,
   - current unselected data,
   - uncurrent selected data,
   - current selected data.

   Form of the data is similar to the valid ANSI standard sequences without the first character ESC (27_{10}). Sequences determining this characters are on the IBM PC type computer named "set graphic rendition".

b) Variable number of records which determine the mapping of the codes, obtained at the scanning of the keyboard (for this purpose the program SCAN.EXE on the ANA system directory is used), to the which can be recognized by the program package.

   This second variable part of the file must contain at least 13 records, i.e the records determining all function keys used at the data input (cf. Section 3.3.1 in the ANA User Manual). Beside this 13 records the file can contain additional 242 records,which can define the function keys or keys representing the ASCII characters.

The following must be considered when creating the key setting definition file:

- The mapping between the scanned key and the code which can be recognized by the program package is realized by four integer values, which are separated at least by one blank. First two values are the values which are obtained by the program SCAN.EXE when pressing the selected key. The second two integer values define the mapping. For function keys the first value is optional, the second is the code of the function key (allowed codes are from 1 to 13). If the second value (the "function key code") is 0, the first value determines the ASCII character (the value is the decimal equivalent of the ASCII character; e.g. for "1" the first value is 49, the second is 0).
- Codes of the function keys are are:

  1 jump to the right to the beginning of the neighboring data,
  2 jump to the left to the beginning of the neighboring data,
  3 jump up to the beginning of the neighboring (upper) data,
  4 jump down to the beginning of the neighboring (lower) data,
  5 data editing termination or (de)selection of the data,
  6 jump to the right for one character,
  7 jump to the left for one character,
  8 backspace (deletion of the character left of the cursor),
  9 delete (deletion of the character on the cursor),
  10 entering mode definition key (insert/overprint toggle mode),
  11 end of the data entering,
  12 jump to the right to the neighboring group of the data (to the beginning of the first),
  13 jump to the left to the neighboring group of the data (to the end of the last),
  -1 operation interruption (return into the interpreter, comment "† operation was interrupted through keyboard" is recordrd in the protocol file, pressing the key in the interpreter causes the exit from the package).

An exemple of the file TASTER.DAT is shown in Fig. 2, which is adapted for EGA graphic card.

```
[1m     color of the record in the normal mode
[1;7m   normal mode -edited data
[4m     color of the record in the select mode
[1;7m   select mode -edited data
0  77 69  1 jump to the right to the beginning of neighboring data
0  75 59  2 jump to the left to the beginning of neighboring data
0  72 63  3 jump up to the beginning of the neighboring (upper) data
0  80 67  4 jump down to the beginning of the neighboring (lower) data
13 28 13  5 data editing termination or (de)selection of the data
0  60 66  6 jump to the right for one character
0  59 65  7 jump to the left for one character
8  14  8  8 backspace -(deletion of the character left of the cursor)
0  83 64  9 delete (deletion of the character on the cursor)
0  82 68 10 entering mode definition key (insert/overprint  TOGGLE mode)
43 78 78 11 end of the data entering
27  1 27 -1 jump to the right to the neighboring groupof the data
           (to the beginning of the first)
0  85 27 12 jump to the left to the neighboring groupof the data
           (to the end of the last)
0  84 27 13 operation interruption (ESC) (return into interpreter,
           consent "loperation was interrupted through keyboard" is recorded
           in the protocol file ,pressing the key in the interpreter causes
           the exit from the package)
48 82 48  0 additional character for zero INS
49 79 49  0 additional character for one  END
50 80 50  0 additional character for two 'arrow down'
51 81 51  0 additional character for three PgDN
52 75 52  0 additional character for four <--
53 76 53  0 additional character for five
54 77 54  0 additional character for six  -->
55 71 55  0 additional character for seven Home
56 72 56  0 additional character for eight 'arrow up'
57 73 57  0 additional character for nine  PgUp
46 83 46  0 additional character for period Del
45 74 45  0 additional character for minus
```

Fig.2: Example of the file TASTER.DAT.

2.3.3. File GRAPH.DAT

Five colors are used by the program package for the plotting of variables on the screen, i.e. the color No.1 for the axes, No. 2 for the first, No.3 for the second, No.4 for the third and No.5 for the fourth variable respectively.

The actual colors which are used for plotting are determined by the file GRAPH.DAT, which is located on the ANA system directory and which must contain at least 5 integer values separated by at least one blank. It means that the color which is used in the package as number I (I = 1, 5)

will be plotted by the color determined by I-th value in the file.

An example of the file GRAPH.DAT is shown in Fig. 3, which is adapted for the EGA graphic card and determines that for coordinate axes yelloow, for the first curve light magenta, for the second curve cyan,for the third curve light red and for the fourth curve light blue color is used.

```
14  13  3  12  9
```

Fig.3. An example of the file GRAPH.DAT

2.3.4. File PLOT.DAT

At the determination of the colors for graphic results documentation by plotter the same way is used as at the plotting on the screen. The only difference is that in the file PLOT.DAT on the ANA system directory the mapping of the program package color I (I = 1, 5) to the consecutive number of the pen is defined. Beside this the name of the plotter address (LPT1 or COM1) is included as the first record in the file PLOT.DAT.

Possible form of the file PLOT.DAT is shown in Fig. 4 showing that for the first color the pen No. 2, for the second the pen No. 1, for the third the pen No. 1, for the fourth the pen No. 3 and for the fifth color the pen No. 4 is used

```
COM1 2  1  1 3   4
```

Fig.4:Example of the file PLOT.DAT.

2.3.5. Command files FORT.BAT and LINK.BAT

Command procedures FORT.BAT and LINK.BAT are used at the operation SSI for compilation and linkage of the new built program for the simulation or optimization.

The current version of the package is adapted to the Microsoft FORTRAN77 V3.31 (compiler) and the corresponding linker (Microsoft (R) 8086 Object linker Version 3.04). Both are used only by the operation SSI in the synthesis branch (all other operations can be performed without compiler and linker respectively).

WARNING:
All other versions of the FORTRAN for IBM PC computers are not yet tested and the operation correctness using them can not be guaranteed.

Compiler and linker are started by the program package indirectly using command procedures FORT.BAT and LINK.BAT, which are located on the ANA system directory. In the case of the compiler or linker relocating or renaming only the mentioned command procedures (FORT.BAT and LINK.BAT) must be modified.

Example of the command procedures FORT.BAT and LINK.BAT form is shown in Figs.5 and 6 respectively, where the compiler and linker location is denoted by C:\F77\.

```
echo off
h:FOR1 %2,%1,obj,NUL,NUL
h:FOR2
```

Fig.5: Example of the command procedure FORT.BAT.

```
echo off
h:link %2,%1,nul,c:\ANA\EXE\unicus+c:\ANA\EXE\analib+c:\ANA\EXE\ssplib
+c:\ANA\EXE\anainp+h:
```

Fig.6: Example of the command procedure LINK.BAT.

2.4. Preparation for the program package start

Before starting the program package the following conditions must be fulfilled for its correct function:

1) The file CONFIG.SYS on the main directory of the medium from which, when the computer is turned on, the operating system is started (boot medium) must contain the record

   DEVICE = \path\ANA.SYS

   where \path\ is the ANA system directory denotation (e.g. C:\ANA\EXE\).

   In this way the screen driver for correct operation of the package is installed; it can remain installed during other applications because it does not disturb other programs.

2) The files TASTER.DAT, GRAPH.DAT, PLOT.DAT and command files FORT.BAT, LINK.BAT respectively on the ANA system directory must be set correctly.

3) On the directory from which the package is started the file ANAASG.DAT, being set correctly, must be located.

4) The following drivers, which are located on the ANA system directory must be started before starting the package:
   a) Keyboard driver KEYB.EXE.
   b) Additional screen driver for the graphic mode; depending on the used graphic interface:
      CGADRV.EXE for CGA graphic interface card,
      EGADRV.EXE for EGA graphic interface card,
      HGCDRV.EXE for HGC graphic interface card.
   c) Printer driver. Because of various possibilities of different printers the default printer driver is named PRNDRV.EXE (defined at the package installation); all other printer drivers are named by the printer type (e.g. FX80 for FX80 printer and compatible).

The contractor performs all necessary preparations for the program package start when installing it, so additional preparations performed by the customer must be done only in the case of the following changes:

- at the package reconfiguration according to the customer wishes the files ANAASG.DAT, TASTER.DAT,PLOT.DAT and command files FORT.BAT, LINK.BAT respectively must be redefined correspondingly (usually only the file ANAASG.DAT must be adapted once for each user, because all other files are set in the agreement with the user at the package installation),
- exchange of the graphic card type requires the start of the corresponding additional screen driver,
- exchange of the printer requires also the start of the corresponding printer driver.

2.5. Start

The appropriate configuration of the program package and all necessary preparations for its start are done by the supplier at the package insta-

llation. The final result of the installation is the command procedure for easy start of the package. An example of such command procedure with e.g. name ANA.BAT is shown in Fig. 7. The function of the command procedure is to start the corresponding drivers (keyboard driver, additional screen driver for EGA graphic card, default printer driver) and at the end also the program package. All started files are located on the directory denoted by C:\ANA\EXE.

```
C:\ANA\EXE\KEYB
C::\ANA\EXE\EGADRV
C::\ANA\EXE\PRNDRV
C::\ANA\EXE\ANA
```

Fig.7: Example of the command procedure for the start of the program package.

After the package installation it is possible to start it by the built command procedure (in the above example by entering the command ANA) from any subdirectory containing the file ANAASG.DAT.

RECOMMENDATION:
The working directory should contain as few files as possible before the package call, this will speed up the work with the package.

2.6. Usage specificities

2.6.1. Peripheral devices addressing

Program package addresses the printer by its system name LPT1; the plotter name is determined by the file PLOT.DAT on the ANA system directory and can be LPT1 or COM1.

For both names all general rules of the operating system are valid; also their parameter (re)definition is possible by the MODE command. This is especially important for the COM1 device!

2.6.2. Screen graphics

Program package is designed to operate with different graphic interfaces (CGA,EGA,HGC) where only the use of HGC in the graphic mode is specific. When using it a tone signal must be waited for and the operation continues after pressing any key. For other two interfaces the recorded instruc-

tions must be followed.

The EGA graphic card is the most suitable for the use of the package because only with it more than two colors can be used what is in some cases needed to attain satisfactory picture clearness.

2.6.3. Screen hardcopy on the printer

All three possible graphic cards (in text, semigraphic and graphic mode) are supported by the corresponding software for screen hardcopy (Print Screen) on a dot matrix printer.

The Print Screen is started by pressing the standard key (e.g. PrtSc). Then the following must be done:

a) for HGC graphic card first the screen mode must be typed in (T for text and semigraphic mode, G for graphic mode of the screen) and in the case of graphic mode also the picture type (L for large and S for small), (Figs. 8 and 9),
b) for EGA and CGA graphic cards in the graphic mode only the picture type has to be determined (L for large and S for small), (Figs. 8 and 9).

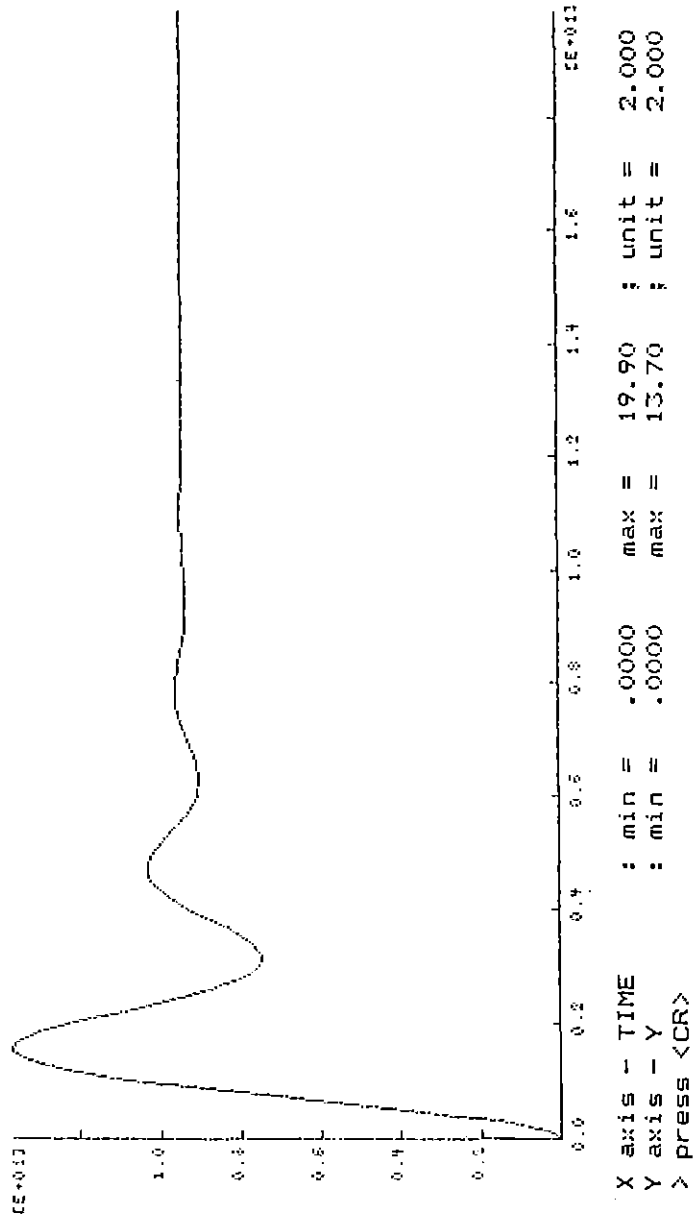


Fig.8: Large picture

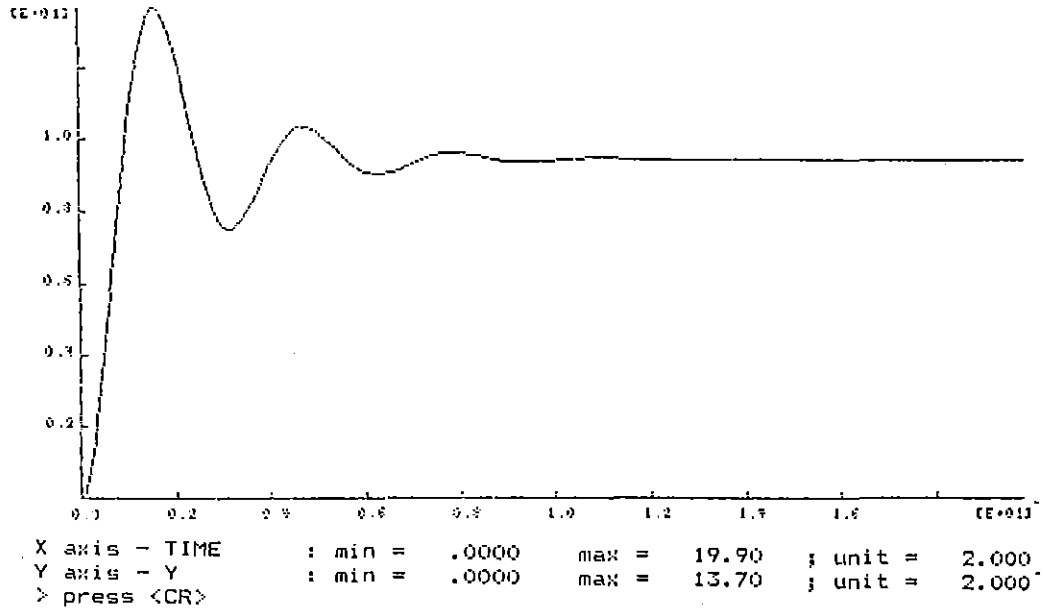


Fig.9: Small picture

2.6.4 Documentation using plotter

Graphic data documentation on the plotter is realized in such way that for any picture a domain on the A4 and A3 format respectively has to be chosen. One of the following five modes can be chosen:

- [1] one picture on A4 format longitudinally,
- [2] two pictures on A4 format upright,
- [3] two pictures on A4 format longitudinally,
- [4] four pictures on A4 format upright,
- [5] format determined by the user (optional on A3).

The domains where the picture can be plotted are illustrated by the corresponding Figures 10 to 16, which were plotted by a plotter and from which it can be seen that except the mode also the following must be defined:

a) using mode [2] the position of the picture (lower, upper),
b) using mode [3] the position of the picture (left, right),
c) using mode [4] the position of the picture (first = upper, second, third or fourth = lower),
d) using mode [5] the position of the left lower point (T_z), the position of the left upper point and the graph orientation (it is recommended that the x -axis span is greater or equal as the y axis span).

Beside this with the modes [1] and [2] the square plotting domain can be defined (the greatest possible square inside the corresponding domain) The choice of the plotting domain is realized interactively and separately for every picture.

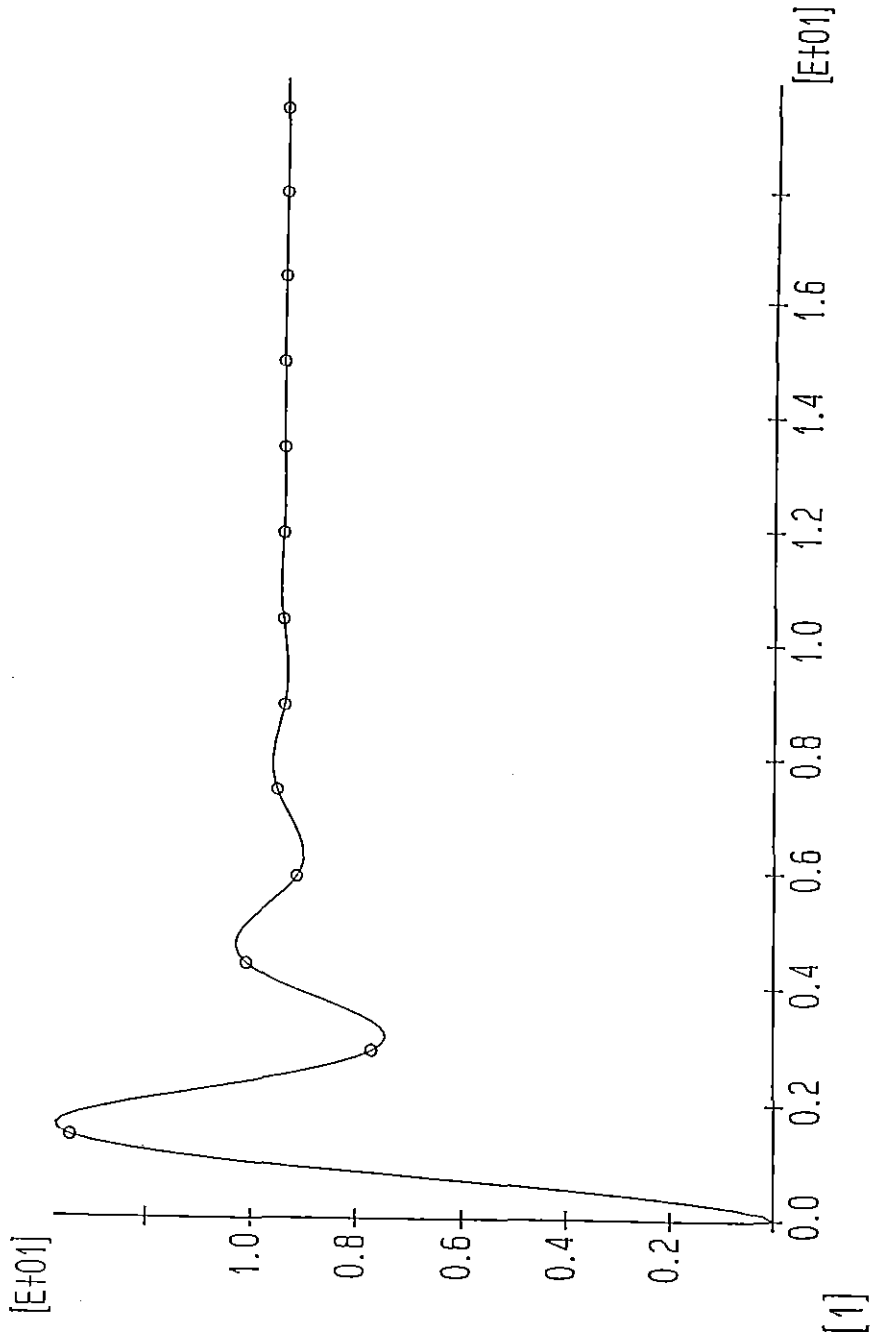


Fig. 10: Mode [1] not square.

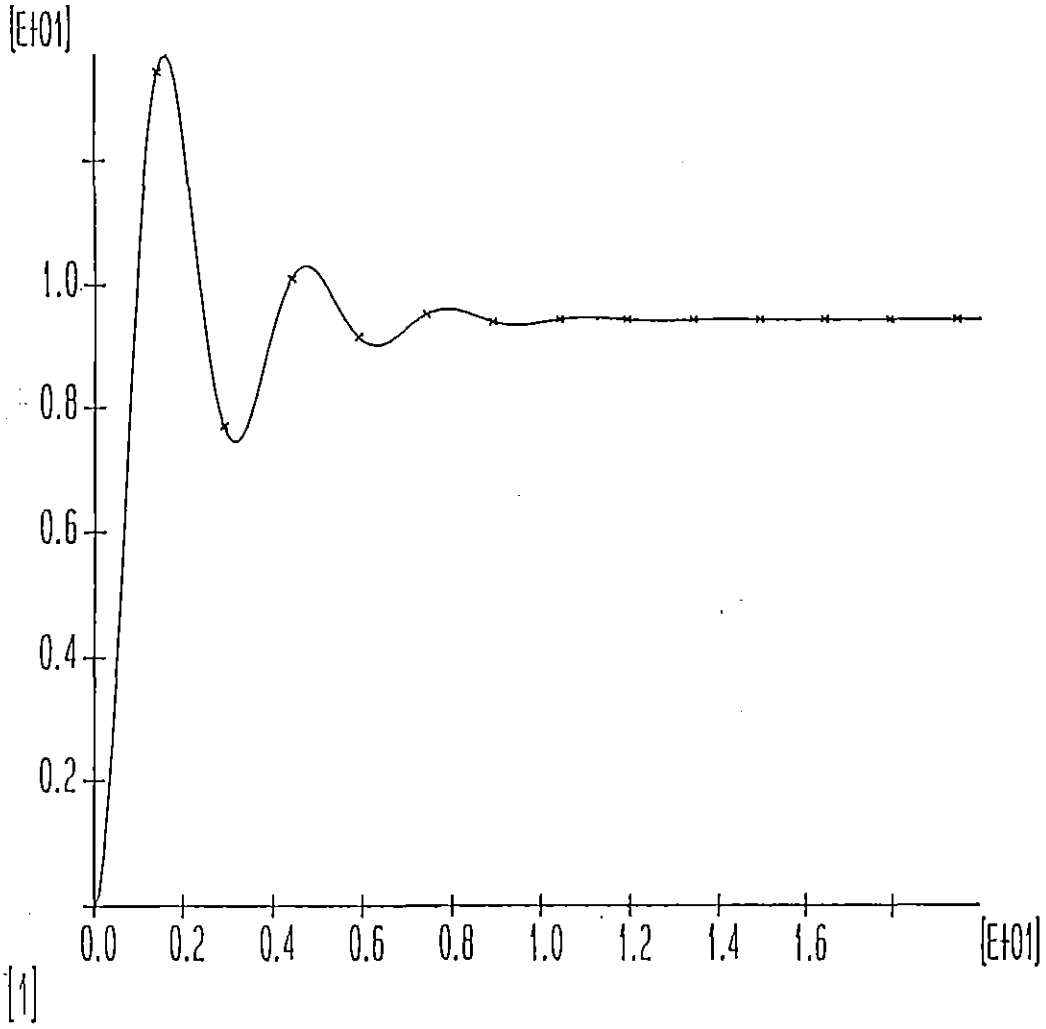
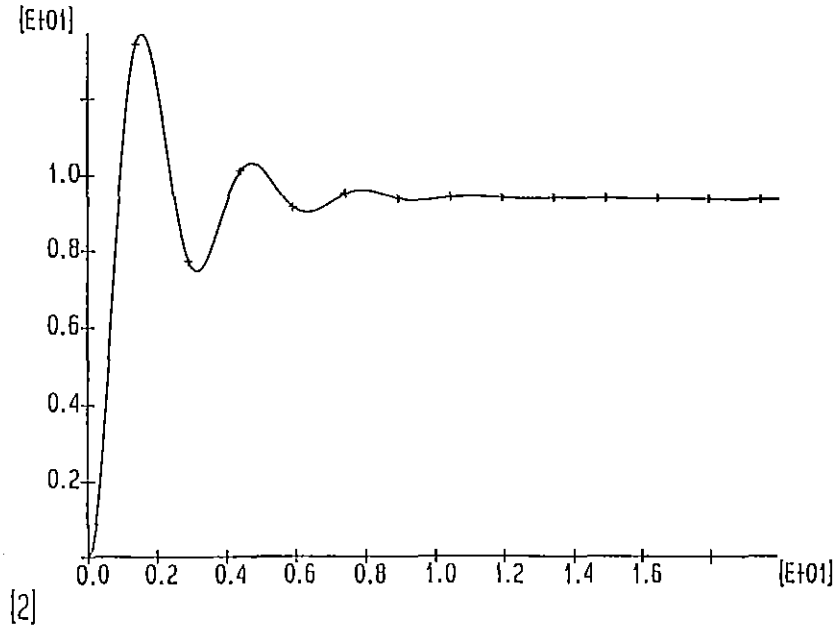


Fig.11: Mode [1] square.



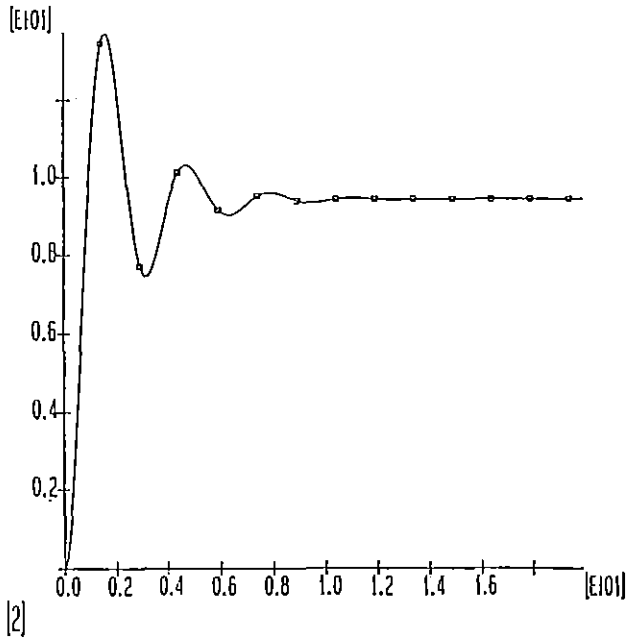
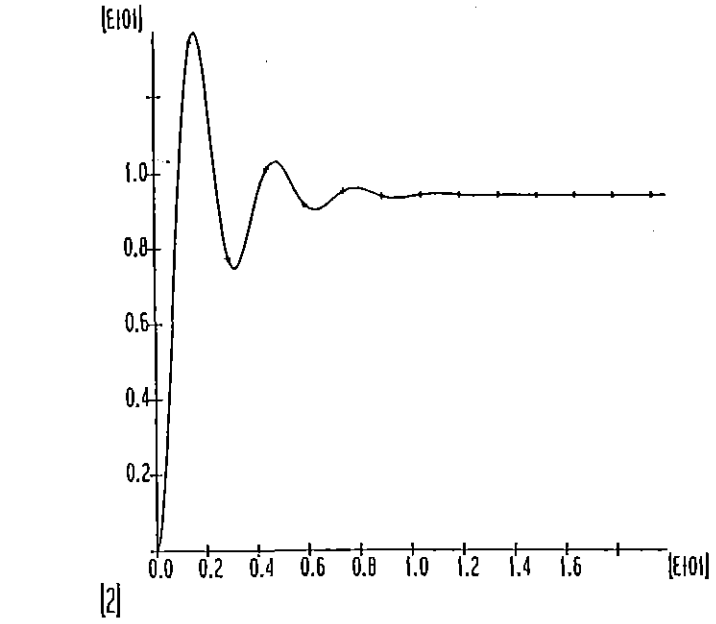


Fig.12: Mode [2] nonsquare (upper) and square (lower).



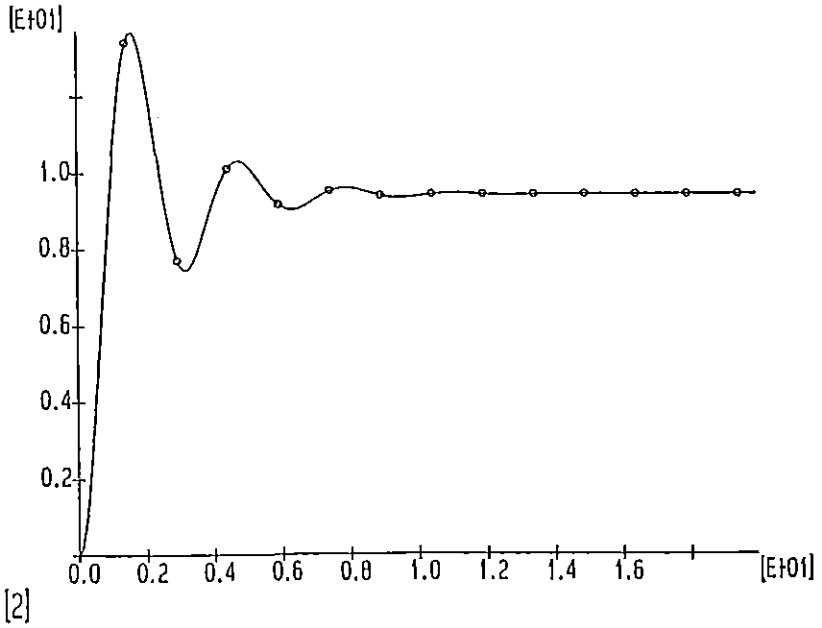


Fig.13: Mode [2] square (upper) and nonsquare (lower).

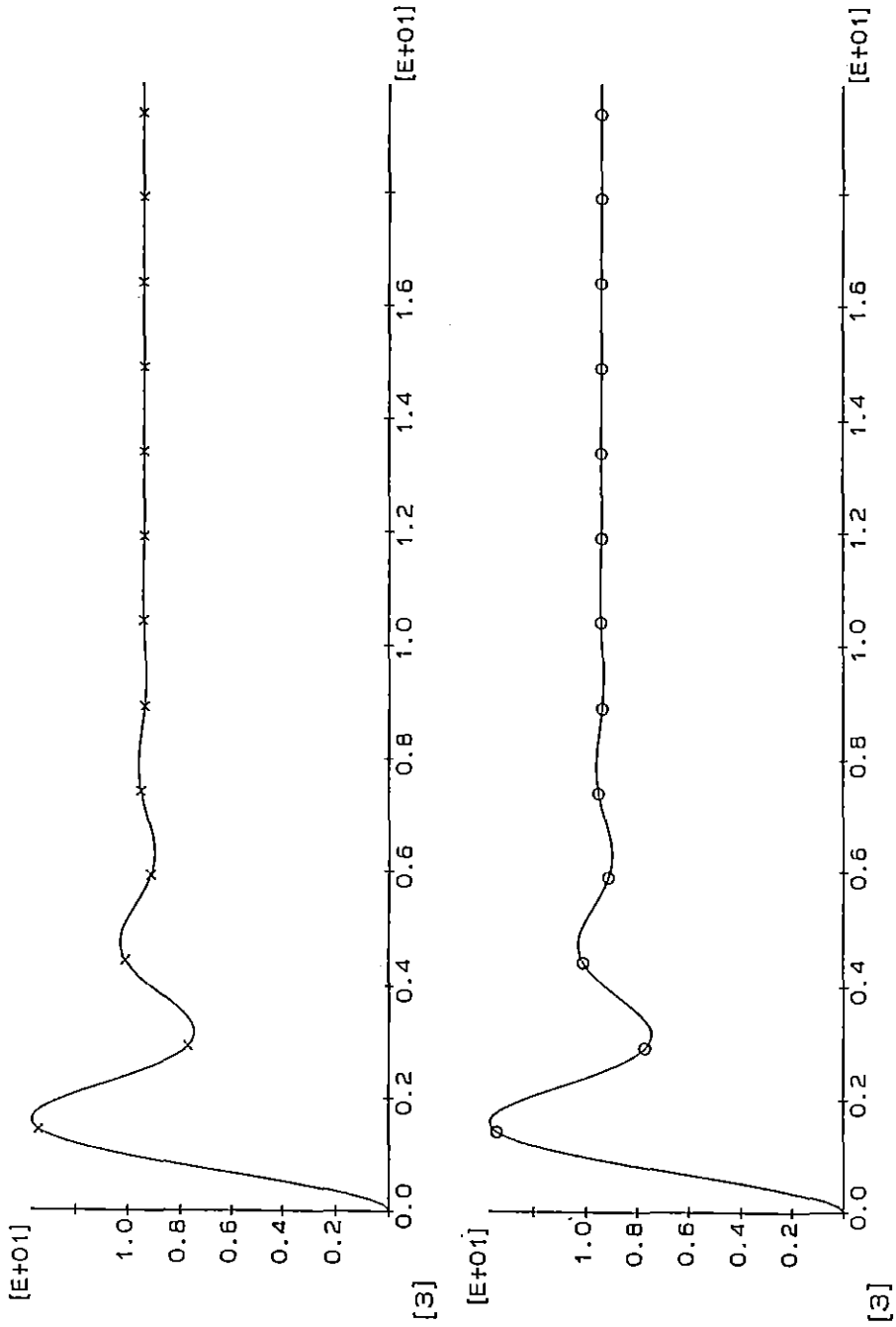
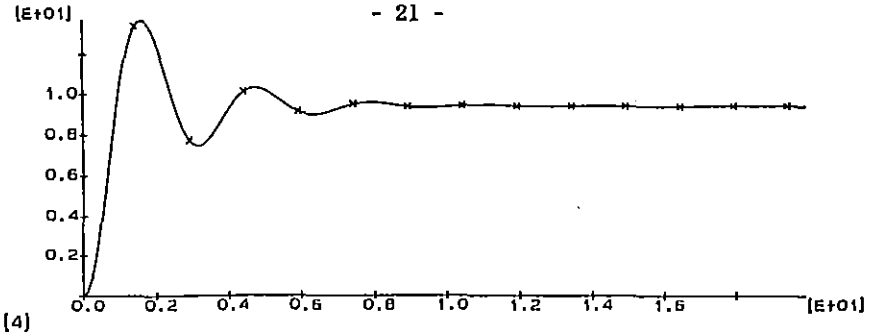
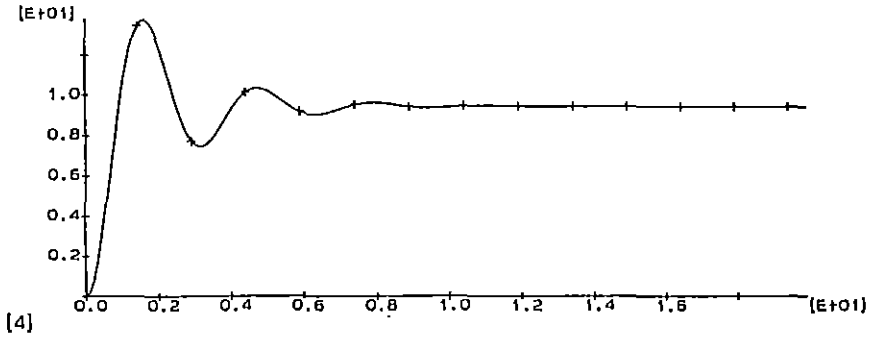

[E+01]
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
[E+01]
[3]
[E+01]
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
[E+01]
[3]

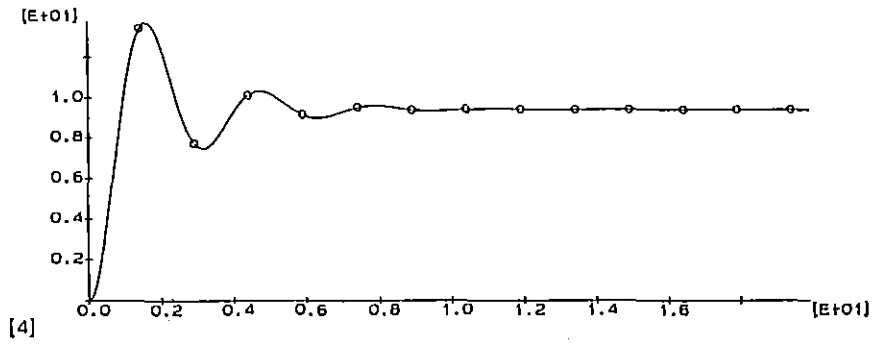
Fig.14: Mode [3].



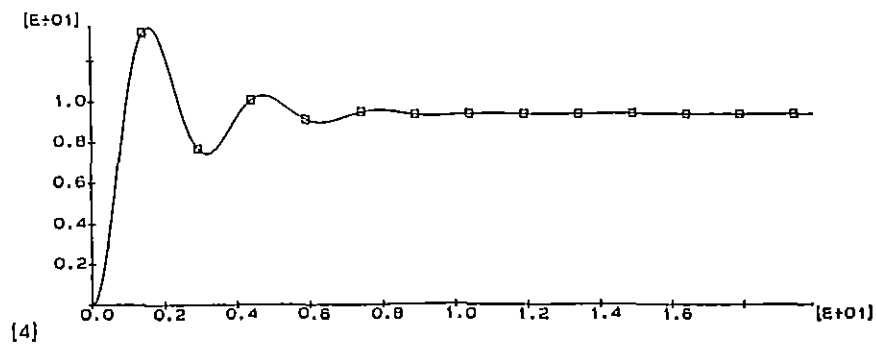
[4]



[4]

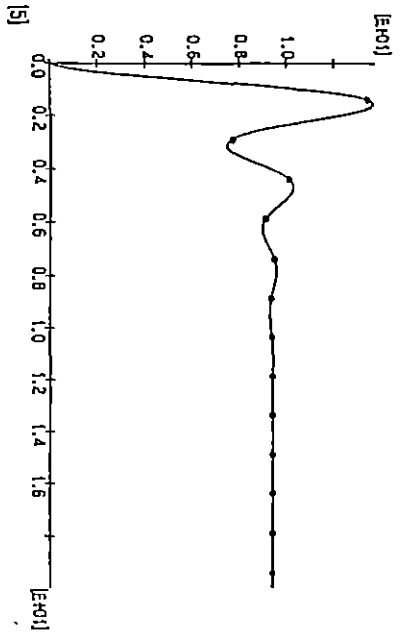


[4]



[4]

Fig.15: Mode [4].


[5]
[E+01]
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
[E+01]

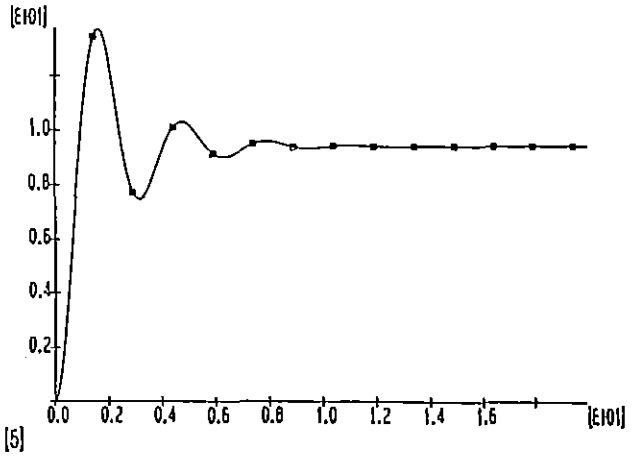

[E101]
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
[E101]
[5]

Fig.16: Mode [5].

2.6.5. User subroutines in the operation SSI

At the SSI operation (simulation of general dynamic structure) in the synthesis branch the subroutines (user blocks TBL, the programs for criterion function calculation and the subroutines for the constraints definition at the optimization) written in the program language FORTRAN can be included. All files containing these subroutines must begin with three records, shown in Fig. 17.

```
$STORAGE:2
$NO STRICT
$D066
```

Fig.17: Necessary records at the beginning of the user files.

2.6.6. Additional remarks

- When using graphic (on the terminal or plotter) the units on the axes are not written if their integer part exceeds 32700.
- All specificities of the particular installation which are not included in this document are the subject of the user and contractor agreement.

3. GENERAL DYNAMIC STRUCTURE SIMULATION

The concept of general dynamic structure simulation represents one of the most complex functions of the CACSD program package ANA. As it has been already mentioned it also enables identification of nonlinear and time-variant processes. Presently, this possibility is based on proper combination of simulation and optimiyation algorithms. In the future, however, we will try to implement some other methods, too.

3.1. Function

The operation has two basic functions:

A) general dynamic structure simulation,

B) off-line optimization of the general dynamic structure parameters.

From the user point of view the realization of one of the stated basic functions for the concrete dynamic structure is performed stepwise in the following order:

a) definition of the structure and structural parameters,
b) definition of the nonstructural parameters,
c) simulation and optimization run respectively,

if the structure was not yet defined, and in the following order:

a) redefinition of the nonstructural parameters,
b) simulation and optimization run respectively,

if the structure has already been defined (at the current or at some of the previous treatment with the package).

The result of the operation for any basic function execution is the internal file SUNICU.TTI (or formatted sequential file SUNICU.DAT) (on the working directory) which contains the simulated time courses of the chosen variables. In the case of the optimization the result is additional formatted sequential file SOPTIM.DAT (on the working directory) which contains the optimal parameter values.

Beside the above mentioned files, which are generated at every simulation or optimization run, the result of the structure and parameters definition (before the first run) is the set of files STRUKT.UN* on the working directory. These files enable the repetition of the runs with redefined nonstructural parameters.

3.2. Method

The simulation of the general dynamic structure and off line optimization of its parameters is realized by a program equipment with the name UNICUS (Universal Computer Simulation) and which consists of:

- UNICUS compiler,
- UNICUS interpreter.

The heart of the UNICUS program is the compiler which builds for every new structure (using the sorting algorithm and exactly defined data base, which is for this structure stored in a file) a new source program for the simulation and optimization respectively.

UNICUS interpreter however is the interface between the user, UNICUS program and the whole program package and ensures a regular function of the whole UNICUS program in a way which is simple and understandable for the user.

A) General dynamic structure simulation

Simulation of a concrete dynamic structure which is defined by the user is block oriented. Blocks are in general multivariable and are mutually conected through inputs and outputs. Every block of the simulation structure can be:

a) Model of the system which is in general dynamic (TGZ, TMZ, CMA, TBL)

Discrete linear models of dynamical systems (TGZ, TMZ) and the models of nondynamical systems (CMA) are results of previously executed operations in the package (in the branches for generation, transformation, analysis and synthesis).
The user block (TBL) which is a subroutine written in FORTRAN and describes the input output behaviour of the concrete subsystem must be built (edited) and tested before the program package start.

b) Signal source

The signal source can be one of standard signals (step, pulse, train of pulses, ramp, sinusoidal signal, Gaussian or pseudorandom binary noise) or it can be given in a file, which is the result of an operation within the program package (internal file with the extension TTI) or in the formatted sequential file which has the same structure as it is required for the internal file TTI generation (description of the operation GENERATION OF TTI).The signal source can be also the user block (TBL) because it can be treated as a system model without any input or a model of the system with arbitrary input.

c) Summing point

In the summing point vector addition or subtraction of the signals is realized.

d) Special block

Special blocks are introduced for easy verification of some more sophisticated but in the control theory often used elements. At the present realization two such elements are possible:
- Kalman observer,
- Kalman filter.

Simulation is performed in such a way that in every discrete moment"k"

(representing the real time moment ($k \cdot T_0$) where T_0 is so called basic sampling time of the structure; the sampling times for discrete linear models TGZ and TMZ) must be the multiples of the basic sampling time) the outputs of all blocks are calculated in corresponding order.

The simulation run duration is given by the user. It can be additionally limited by a prescribed maximum of calculated points and/or by a logic condition for simulation run termination.

The variables which should be written in a file and the way and form of the recording can be adapted to the concrete simulation run.

B) Off-line optimization of the general simulation structure parameters

Off-line optimization of the general simulation structure parameters is based on already described general dynamic structure simulation. For each simulation run (for concrete fixed values of parameters to be optimized) the value of the criterion function is calculated and used by an implemented searching optimization algorithm for further optimization procedure.

For the given initial conditions the optimum (minimum or maximum) is approached iteratively on the way determined by the optimization algorithm considering criterion function values. The values of the optimized parameters, which are the result of one optimization run represent in general a local optimum, because the procedure depends very much on the shape of criterion function, on the initial conditions and conditions which terminate the optimization run. The decision whether or not the obtained local optimum is also the global one must be performed by the user himself.

For each simulation run, which is executed inside the optimization run, all the characteristics are valid which are described for simulation. The only exception is the tecording which is performed only at the end of the optimization procedure (for the optimal parameter values). Implemented searching optimization algorithm is successful for the optimization of several independent variables with arbitrary constraints. The criterion function can be either the implemented general quadratic criterion or a subroutine written by the user in FORTRAN. Constraints can be implemented as logic conditions or by user written FORTRAN subroutine.

For each optimization run it is necessary to define initial values of optimized parameters and initial search step. The optimization procedure termination is determined by one of three conditions (number of allowed iterations, minimal change of the criterion function or minimal change of step between two successive iterations). If required the parameters which enables an estimate of optimization performance can be written on the output communication device. Optimal parameter values are at the end of the optimization run always written in a file.

Remarks:

Due to the complexity of the whole UNICUS program in the remarks detailed descriptions of those segments are given, which are from the user point of view of great importance for easy, simple and successfull work.

a) UNICUS interpreter

UNICUS interpreter ensures the correct execution of the whole UNICUS program. In the case of a new simulation structure it enables the following basic functions:

- Simple simulation scheme definition.
- Simple structural parameters definition, i.e. parameters which can not be altered witout a new start of the UNICUS compiler (which would build a new source program for simulation and optimization respectively).
- Simple definition of nonstructural parameters i.e.parameters which can be altered without changing the already built (linked) program version for simulation and optimization respectively.
- The guidance of the user through above stated functions interactively, on line testing of the entered data correctness and in the case of errors redefinition of the normal communication course. If an error occurs, which can be corrected by the new input, such input is enabled, otherwise the definition of basic parameters for this operation is terminated and the work course is continued by a new operation choice in the synthesis branch.
- Generation of the corresponding data base (file) which is used by UNICUS compiler to build a new source program if the structure is correct in the sense of errors, which can be foreseen in advance.
- Renaming of files containing structural data (STRUKT.UN*) and starting the UNICUS compiler, FORTRAN compiler, LINKER and at the end also star-

ting of the already linked program (first simulation or optimization run is executed).

If the simulation structure is already defined (repetition of the simulation and optimization runs respectively for the structure defined by the set of files *.UN* with the same name and different extensions) the functions of the UNICUS interpreter are as follows:

- enables the corresponding structure choice (file *.UNI),
- plots the simulation scheme on the screen,
- enables simple redefinition of nonstructural parameters (and changes the data in files *.UN*),
- guides the user through redefinition of nonstructural parameters, tests the correctness of entered data and enables the treatment of eventual errors in the already described way,
- starts a new simulation and optimization run respectively if the parameter redefinition was not interrupted.

From the above described functions of the UNICUS interpreter and from the fact that all programs built by UNICUS return the program flow supervision to a new operation choice in the synthesis branch, it is obvious, that the whole SSI operation (except the definition of the scheme and parameters) is executed automatically without any intervention of the user.

b) Definition of the simulation scheme

Interactive definition of the simulation scheme is performed in one or two steps where the lower part of the screen is used for the communication between the package and the user and the upper part of the screen for the simulation scheme presentation.

At the beginning of the first step the maximal general scheme determined in advance is plotted in the upper left part of the screen (Fig. 18). It includes the block with the name process (PR), controllers in the direct (R1) and feedback (R2) path, input noise filter (GI), output noise filter (GO), reference signal filter (R3), feedforward controller (R4), parameter estimator (ES), state observerr (OB), and special block which joins the control (UR) and nocontrol (UN) input signals into the one control vector (UU).

In the first step of the simulation structure definition the accommodation of the scheme in Fig. 18 to the concrete example is realized. During

the accommodation the type of the block shown on the screen in the inverse mode should be defined (in Fig. 18 this is block PR). As the type of the block may influence the outlook of the simulation scheme, the order of the block type definition is determined by UNICUS interpreter which currently adapts also the picture on the screen.

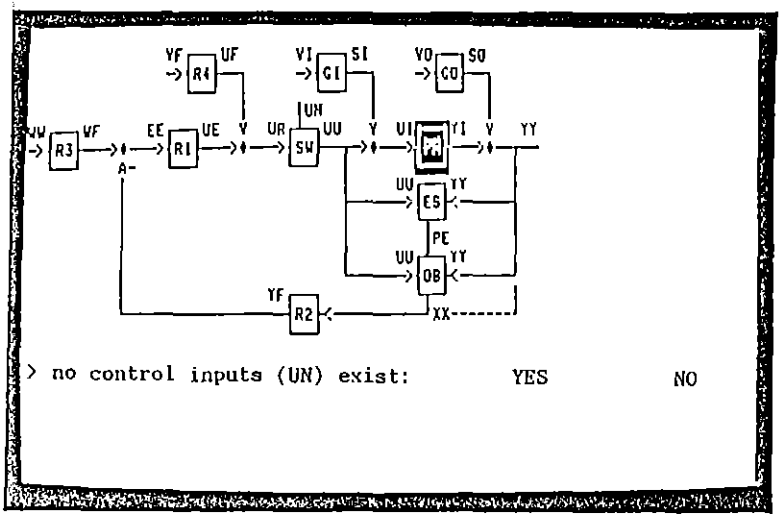


Fig.18: Picture of the screen at the beginning of the simulation scheme definition.

While defining the type of the block PR it should be defined whether or not the process model contains no control inputs. This means that the block PR is always represented by one of the possible system representations (TGZ, TMZ or TBL) and that the block SW exists only if the PR has no control inputs.

The definition of the block types for GI, GO, R1, R2, R3 and R4 requires that the type of the concrete block is defined by the following abbreviations:

BL: means that the block is preserved (BLock) and is represented by the one of possible system models (TGZ, TMZ, CMA or TBL). This block type does not cause any change on the picture of the simulation scheme.

DC: determines that the input of the concrete block is connected directly to its output (Direct Connection). This block type causes that instead of the rectangle a direct connection from the input to the

output is plotted on the screen and that some of the variables are redefined.

OC: determines that there is no connection between the input and the output of the concrete block (Open Connection). This block type causes usually greater changes in the simulation scheme. It namely means that the concrete signal flow path is cut off and so also some other blocks may be superfluous (senseless). This abbreviation is the only one which causes the order of the scheme determination in the first step to be changed.

For the blocks ES and OB it must be defined through the type whether or not they are included into the simulation scheme.

If it is impossible to determine the required simulation scheme by the adjustment of the scheme in Fig.1 the second step must be undertaken, i. e. the scheme completion by additional elements.

The maximal structure of additional element is shown in Fig. 19. For each such element first the signal flow, i.e. the name of the signal representing its input and the name of the signal (in some cases also the position) where the output of the additional element is incorporated, must be defined.

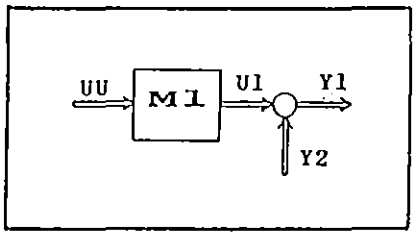


Fig.19: Maximal form of the first additional element for the simulation structure completion in the second step.

After the definition of the signal flow the maximal form of the additional element is plotted on the upper right part of the screen and if necessary also some modification on the already determined main scheme are performed, which are the consequence of the additional element incorporation (variable redefinition). The incorporation of the additional element into the simulation scheme is uniquely determined by two markers (serial number of the additional element on the inverse background). The first is

on the additional element and the second is on the position of its inclusion.

For each additional element the block type (possible types are BL, DC and OC) and the type of the summing point (addition, subtraction; in some cases the additional elements has no summing point) must be determined at the end. This may also modify the additional element image on the screen.

The procedure of additional elements including can be repeated until the desired simulation scheme is obtained.

The following comments should be added for the simulation scheme definition:

- UNICUS compiler enables the simulation of the general dynamic structures without any limitations on the number of blocks (conceptual limit represents the size of the program which is possible to be run on a concrete computer). The UNICUS interpreter realization however limits the number of blocks represented by possible system representations (TGZ, TMZ, CMA and TBL). Beside the blocks on the basic scheme (Fig. 18) up to 8 additional elements according to Fig. 19 are allowed.
- First four additional elements are plotted one under the other in the right upper part of the screen. The same is true also for the second four additional elements; however they overlap the first four ones, due to the lack of the space.
- The markers denoting the positions of the additional elements inclusion into the whole structure enable unique and evident representation of signal flows. The lucidity problems (marker overlapping) may appear only if several additional elements should be included in the same position and they are not defined in the right turn (in the turn of the signal flow).

c) First simulation run

After the determination of the simulation scheme the UNICUS interpreter guides the definition of the structural and nonstructural parameters. Because they are in the mixed order and because some of structural parameters are defined indirectly, subsequently the particular parameter type and the way of its determination will be denoted separately. So the procedure of parameter redefinition and the fatal (operation terminating) error description will be more evident.

The determination of the parameters is performed in the following order:

- parameters of the blocks representing the models of the corresponding subsystems models (blocks PR, GI, GO, R1, R2, R3, R4, M1, ..., M8),
- parameters of the blocks representing special control elements (blocks ES and OB),
- parameters of the signal sources,
- parameters in connection with the printing of the variables,
- parameters determining the duration time of the simulation run.

For each block representing the model of the concrete subsystem the following data must be typed in:

- model type (TGZ, TMZ, CMA or TBL),
- name of the file containing the data (files *.TGZ, *.TMZ, *.CMA, *.TBL),

if the model type is TMZ also

- initial state vector $\underline{x}(0)$,

and if the model type is TBL also:

- dimension of the input vector (M),
- dimension of the no control input vector(MN) if the block represents the process,
- dimension of the output vector (L),
- sampling (executing) time of the block (T0) if the block represents the process,
- the form of the users block call statement in the program (considering actual names of inputs, outputs and other variables),
- dimension of the state vector (MSS),
- dimension of the parameter vector (MPA).

All parameters considering the definition of the model TBL are structural parameters, except the sampling time.

For linear models TGZ, TMZ and CMA the model type is structural, while the file name containing the model data is nonstructural parameter. For models of TMZ type the initial state vector is nonstructural parameter. For linear models TGZ, TMZ and CMA the structural parameters (M, MN, L, MSS and MPA) are determined indirectly on the basis of the data in the chosen file. Basic sampling time of the structure is nonstructural para-

meter and is determined by the sampling time of the process. If the process is linear dynamic system (TGZ, TMZ) the basic sampling time of the structure is defined indirectly on the basis of the process sampling time (defined in the file of the process), otherwise it is determined by the user at the definition of the user block (TBL) parameters.

Block representing estimator can be in the present version only user defined block (type TBL) and for it the following structural parameters should be defined:

- the name of the file containing the "model" of the estimator (*.TBL),
- dimension of the estimated parameters vector (dimension of the output vector), if it is not yet defined by the input of some previously determined blocks,
- the form of the users block call statement in the program,
- dimension of the state vector (MSS),
- dimension of the parameter vector (MPA).

For the observer block the defined type can be Kalman observer, Kalman filter or user defined block. The observer type is structural parameter.

For the user block (TBL) type observer the following structural parameters should be defined:

- the name of the file containing the "model" of the observer (*.TBL),
- dimension of the estimated state vector (dimension of the output vector), if it is not yet defined by the input of some previously determined blocks,
- the form of the users block call statement in the program,
- dimension of the state vector (MSS),
- dimension of the parameter vector (MPA).

Simulation, entering of parameters, their notations and the definition of the parameters for Kalman observer and Kalman filter is based on the description of the operation KALMAN OBSERVER in the branch for observer synthesis.

For the Kalman observer definition the UNICUS interpreter requires the following:

- name of the file containing the process model used for the state estimation (*.TMZ),
- dimensions of particular vectors (free vector (MD) and measurable dis-

turbances vector (MF)) inside the no control input vector of the process (MN = MD + MF + MZ)
- initial vector of the state prediction $\underline{x}_p(0)$,
- name of the file containing the Kalman observer matrix (*.CMA) (the result of the operation KALMAN OBSERVER).

All above stated parameters except those dealing with no control input decomposition are nonstructural parameters of the Kalman observer. Structural parameter, which is determined indirectly by the model of the process, is of course the dimension of the observer output vector, which is equal to the number of states of the process model used in the Kalman observer.

For the Kalman filer the following must be defined:

- the kind of process model which is used for the state estimation (time invariant or time variant model),
- name of the file containing the process model (*.TMZ) (in the case of time variant process model this file contains only the initial values of the model matrices A(0), B(0), C(0) and D(0)=0)
- dimensions of particular vectors (free vector MD and vector of measurable disturbances MF) inside the vector of no control inputs (MN=MD+MF+MZ),
- initial vector of the state prediction ($\underline{x}_p(0)$),
- covariance matrix of the input noise (R_Z),
- covariance matrix of the output noise (R_N),
- initial value of the covariance matrix of the state prediction error ($P_p(0)$).

Again all data are nonstructural parameters except the data about process model kind and the data about the dimensions of vectors inside the no control input vector. The dimension of the observer output vector (structural parameter) is determined indirectly by the process model.

All signal sources parameters are nonstructural (all structural parameters of the sources, i.e. the number of signals contained in a particular source, are determined implicitly by the dimensions of inputs and outputs of used and up to the actual moment already defined models). Each signal of a concrete source can be one of standard signals (step, ramp, pulse, train of pulses, sinusoidal signal, Gaussian or pseudorandom binary noise). Some of the source signals may be given also by the corresponding

files (nonstandard signals). Here the file can be either a file with internal data structure (extension of the file name is TTI) or a formatted sequential file, where the signals are written in the same way as it is required for the internal file TTI generation (the description of the new user file generation at the operation GENERATION TTI). The following must be considered at the use of nonstandard signals:

- If more signals of a single source are desired to be of nonstandard form, all their values (samples) must be stored in the same file.
- At most three different sources may contain nonstandard signals.
- In the file *.TTI enough data must be contained for at least 20 samples of each variable calculation.

Similarly with the sources also all parameters determining the form of the output, the way of its execution and the recorded variables are nonstructural parameters. The recorded parameters are:

- the type of the file, where the time series of the chosen variables should be written (formatted sequential file SUNICU.DAT which has the same form as it is required at the generation of the new user file TTI or a file in the internal data structure of the package SUNICU.TTI),
- the number and the names of variables to be recorded (only inputs and outputs of the simulation scheme blocks can be recorded),
- parameter (NO) which determines that only every NO-th computed sample is written in the file. So it defines the sampling time for this file (NO-times greater than the basic sampling time of the structure).

The resulting file type choice is realized with an intention that the resulting time series may be used easily also outside the package. However the formatted sequential file is useful also in the cases where the internal data file constraints for files *.TTI would be too rigorous (the maximal number of samples is 401 and maximal number of signals is 20). For the formatted sequential file SUNICU.DAT no constraints exist.

Parameters which determine the duration time of the simulation run are:

- The time of the system observation start (T_s) is the time (relatively to the time T=0sec where the simulation starts) where we begin to observe the system (e.g. start of the variables printing). In the case where the writing in the internal data file is desired it influences, in connection with the basic sampling time of the structure (TO) and the parameter (NO), the possible simulation run duration time (T_f). In

any case the simulation run duration time must be greater than the time of the system observation start.

- The duration time of the simulation run (T_f) is the time (relatively to the T=0sec where the simulation starts) where the simulation is to be terminated.
- Additional condition for the simulation run termination is the maximal number of the computed samples (NC).
- Additional condition for the simulation run termination can be defined also with a FORTRAN logic condition (the simulation is terminated when it obtains value TRUE).

All parameters determining the simulation run duration except the FORTRAN logic condition are nonstructural parameters.

d) First optimization run

The sorting and the definition of structural and nonstructural parameters for off line optimization of the concrete dynamic structure parameters is the same as it would be for the definition of the simulation of the same structure. The only difference is that the sequences defining the optimization option of the UNICUS program are added (inserted) between the definition of the sources and parameters respectively.

The sequence determining the structural and nonstructural optimization parameters consists of several groups of data which define in turn:

- parameters to be optimized,
- initial conditions for the optimization run,
- criterion function and constraints,
- conditions for the optimization run termination and output.

With UNICUS program it is possible to optimize up to 10 parameters and each one of them must be an element of a parameter vector of the used model (TGZ, TMZ, CMA or TBL), where optimized parameters may belong to different models.

For the parameters to be optimized the following must be defined:

- the model it belongs to (indirectly through the block name),
- the index of the parameter in the parameter array of the model.

The optimized parameters are in turn of their definition joined into the vector of optimized parameters.

All parameters determining the optimized parameters are structural. Special care should be taken at the definition of the optimized parameters index in the array of the linear model parameters (TGZ, TMZ and CMA) because this index is determined by the UNICUS program uniquely (see pt. f of this remarks).

The parameters which determine the initial conditions for the optimization run are nonstructural and are defined for each optimized parameter by:

- initial value of the optimized parameter,
- initial search step.

Initial value of the optimized parameters vector depends on concrete problem and no additional conditions exist, except in the cases where the constraints are included in the optimization. In such cases the initial value of the optimized parameters vector must be such that no constraint is violated.

The initial search step should be chosen in such a way that the optimum lies somewhere in the neighborhood of the straight line determined by the vector of optimized parameters and the initial search step. If even an approximate position of the optimum is not known, the initial search vector should be in the rank of 10% of the initial value of the optimized parameters vector with all nonzero elements (the optimization algorithm internally scales the quantities with the given initial value of the search step vector).

In structural definition it may be chosen that the criterion function is the implemented general quadratic criterion or a subroutine written by the user in FORTRAN.

If the criterion function is a subroutine, it is uniquely defined with the name of the file containing this subroutine and with the form of the subroutine call (it is adapted to the concrete structure, i.e. to the names of the variables). Both parameters are structural.

If the implemented general quadratic criterion is used:

$$\underline{y} = \underline{x}^T(0)\cdot Q_0\cdot\underline{x}(0) + \frac{1}{N}\sum_{k=1}^{N}(\underline{x}^T(k)\cdot Q\cdot\underline{x}(k)) + \underline{x}^T(N)\cdot Q_f\cdot\underline{x}(N) + \\ + \underline{y}^T(0)\cdot R_0\cdot\underline{y}(0) + \frac{1}{N}\sum_{k=1}^{N}(\underline{y}^T(k)\cdot R\cdot\underline{y}(k) + \underline{y}^T(N)\cdot R_f\cdot\underline{y}(N)\ , \qquad (1)$$

where the vectors $\underline{x}$ and $\underline{y}$ are arbitrary variables (input or output of a block) in the simulation scheme and the matrices Q_0, Q, Q_f, R_0, R, R_f are the positive semidefinite weighting matrices, the following must be defined:

- the names of the variables representing vectors $\underline{x}$ and $\underline{y}$ in the criterion (1),
- the matrices Q_0, Q, Q_f, R_0, R, R_f in turn.

The names of the variables for the general quadratic criterion are structural and all weighting matrices (their elements) are nonstructural parameters. The criterion (1) is calculated in the time interval from the beginning of the system observation (T_s). (T_s is defined by the parameters for simulated variables recording).

Arbitrary constraints can be included in the optimization. They are defined by a subroutine in FORTRAN or with logic conditions. If a subroutine is used, it is defined by the name of the file and with the form of the subroutine call. Logic conditions must be written in the form of FORTRAN logic condition (constraints are violated if a logical condition becomes .TRUE.; up to 10 such conditions may be included). All parameters determining the constraints are structural parameters.

The optimization parameters, the optimization termination parameters and the recording parameters are:

definition of the optimum type (minimum or maximum),
- condition for the optimization run termination (criterion function variation, normalized distance between two successive optimized points, maximal number of iterations),
- conditions for the recording of the current optimization parameters.

The parameter defining the optimum type is structural parameter, all other are nonstructural.

Because the "unit" in the optimization run is one iteration (including several calculations of the criterion function and several simulation runs respectively), the optimization termination conditions are tested at the end of every iteration. The optimization is terminated:

- if the difference (variation) of the criterion function value between the current and the previous iteration is less than the given one,

- if the normalized value of the distance between the current and the previous optimal point is less than the given one,
- if the successive number of the iteration is equal to the given maximal number of iterations.

In the case of optimization it is possible to define that the current optimization parameters are written on the output communication device: at the end of each iteration (only if this device is the terminal screen), only at the end of the optimization run or they are not written at all. The current optimization parameters, which can be written are:

- successive number of the executed iteration,
- number of criterion function calculations,
- normalized distance between the optimal points of the current and the previous iteration (DS),
- the difference of the criterion function value of the current and the previous iteration (DF),
- the current "optimal" value of the criterion function (FK),
- the current "optimal" vector of the optimized parameters.

Regardless to the described recording at the end of the optimization run the optimal parameter vector is written in the file SOPTIM.DAT and the time courses of the variables defined by the parameters for the recording definition (see pt. c of this remarks) are stored in the file SUNICU.TTI or SUNICU.DAT.

e) Repetition of the simulation or optimization runs

If the simulation or optimization scheme have been already defined in one of the previous treatments with the package (i.e. if the first simulation and optimization run respectively has been already performed), the concrete treatment can be repeated with changed or unchanged nonstructural parameters without new compilation with the UNICUS compiler. The latter is much faster than the definition and the execution of the first run of the treated operation.

The redefinition of the nonstructural parameters for the chosen simulation structure (if needed) is conducted by UNICUS interpreter in the following order:

- basic sampling time of the structure
- parameters connected with the recording of the variables

- parameters determining the simulation run duration time,
- parameters of the blocks representing the corresponding subsystem model (blocks PR, GI, GO, R1, R2, R3, R4, M1, ... , M8),
- parameters of the Kalman filter and Kalman observer respectively,
- parameters of the signal sources.

If the optimization is defined with the structure also the following must be redefined:

- initial conditions,
- conditions for termination and recording,
- weighting matrices if the implemented generalized quadratic criterion is used.

f) Vector of the parameters for linear models (TGZ, TMZ,CMA)

The description of the parameter vector for linear models (TGZ, TMZ and CMA) may be important for the definition of their optimized parameters and for the problems which require better knowledge about the simulation and optimization program structure respectively. (e.g. if the user himself writes the parameter estimation program).

In the discrete model parameter vector of the dynamic system, which is represented in the state space (TMZ), all the model matrices are joined. The matrices in the parameter vector are placed in the order A, B, C and D; each of them is represented in the vector columnwise. The index of the concrete parameter vector is thus defined by:

$$i = iv + (is - 1)*IV + ioff \qquad (2)$$

where: iv - the row of the matrix where the element is placed,

is - the column of the matrix where the element is placed,

IV - number of all rows,

$$ioff = \begin{cases} 0 & \text{- for elements of matrix A} \\ N^2 & \text{- for elements of matrix B} \\ N(N+M) & \text{- for elements of matrix C} \\ N(N+M+L) & \text{- for elements of matrix D.} \end{cases}$$

The dimension of the entire parameter vector is [N*(N+M+L)+M*L]

For the discrete models of the nondynamic systems (CMA) the vector of parameters is determined in the same way as for dynamic systems. It

should be considered that the "order" of such systems is equal one (N=1) and that only the D matrix is important (indexes of this matrix elements in the parameter vector are equal i=(1+M+L)+1,...,(1+M+L)+L*M).

In the discrete model parameter vector, written in the frequency domain (TGZ) all the coefficients of transfer functions numerators and denominators are joined. (if the transfer functions in the transfer function matrix are in the factorized form operation SSI converts them in the polynomial form immediately after reading them from the internal data base and stores them in the parameter vector of the block). The coefficients of the numerators and denominators are placed in groups in turn of the transfer functions G_{11}, G_{21}, G_{31},...,G_{L1}, G_{12},..., G_{LM} and G_0 for multivariable systems. For the univariable systems G_0 is on the first place, because it is the only transfer function.

The parameters of a particular transfer function are determined on the basis of the modified internal data structure into the following form:

$$G(z) = \frac{b_1 z^{-ND} + \cdots + b_{NS+1} z^{-(ND+NS)}}{1 + a_2 z^{-1} + \cdots + a_{NI+1} z^{-NI}} \qquad b_1 \neq 0 \quad . \qquad (3)$$

If for representation (3) the "transfer function order" is defined according to:

$$NP = \max(NI, NS+ND+1) \quad , \qquad (4)$$

the parameter vector which defines the transfer function coefficients uniformly is given by the Table (1).

Table 1: The position of the transfer function numerator and denominator coefficients in the parameter vector

<- denominator ->				<---------- numerator ---------->					
1	2	...	NP	NP+1	...	NP+ND	NP+NZ+1	...	2NP
a_2	a_3	...	a_{NP+1}	0	...	0	b_1	...	b_{NP-ND}

$a_i = 0$ for $i > NI+1$

$b_i = 0$ for $i > NS+1$

The index of a particular coefficient of the transfer function can be easily determined by the aid of Table 1 and described order of coefficients groups (for the multivariable systems the corresponding offset due

to Table 1 should be added to the double sum of the orders of the transfer functions which lie before the chosen transfer function; for the univariable systems the index of the corresponding element is determined by Table (1)).

The dimension of the entire parameter vector for multivariable systems is equal to $\left[2\cdot\left(\sum_{i,j} NP_{Gij}+NP_{G0}\right)\right]$ and for univariable systems $[2\cdot NP_{G0}]$ respectively.

When composing the parameter vector for the block with the name process (PR) the following modification should be taken into account:

1) If the system is univariable or if the common transfer function (G_0) of the multivariable system has a delay, the delay of the transfer function (3) is reduced for one unit against the actual delay.
2) If the common transfer function of the multivariable system (G_0) has no delay, the actual delay of all particular transfer functions (G_{ij}, i=1,L, j=1,M) is reduced for one unit and such delays are then taken into account for the parameter vector determination.

As an example the multivariable model not representing a block with the name process (PR) is given:

$$\frac{1}{1+3z^{-1}+2z^{-2}}\cdot z^{-2}\cdot\begin{bmatrix}\dfrac{1}{1-3z^{-1}} & \dfrac{1+z^{-1}}{1+5z^{-3}}\cdot z^{-1}\\ \dfrac{1-2z^{-1}}{1-4z^{-1}} & 4\end{bmatrix} \qquad (5)$$

Because:

$NP_{G11} = \max(1,0+0+1) = 1$
$NP_{G21} = \max(1,1+0+1) = 2$
$NP_{G12} = \max(3,1+1+1) = 3$
$NP_{G22} = \max(0,0+0+1) = 1$
$NP_{G0} = \max(2,0+2+1) = 3,$

the parameter vector is:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-3	1	-4	0	1	-2	0	0	5	0	1	1	0	4	3	2	0	0	0	1
G_{11}			G_{21}				G_{12}						G_{22}				G_0		

g) User subroutines

The user can easily "extend" the capabilities of the UNICUS program by including his own subroutines into the simulation and optimization respectively. Here some program characteristics implemented by the UNICUS compiler should be considered (it is supposed that the user is familiar with program the language FORTRAN, because all subroutines should be written in it).

The inclusion of subroutines, which must be functionally tested prior to the package entering, is simple. All conditions for the correct inclusion of the subroutine are fulfilled by the definition of the file where the source code of the subroutine is stored and by the description of the subroutine call (the form adapted to concrete variable names for defined structure). If necessary also the definition of the corresponding arrays which should be additionally reserved in the new linked program (the UNICUS interpreter takes care for the compilation of the subroutine and for linking) must be added. The file containing the source code of the user subroutine may contain several subroutines. If several user blocks are included into the simulation scheme, each of them must be stored in a separated file and each subroutine can be defined only in one user block (if e.g. two functionally equal user blocks should be included into the structure, "two" subroutines should be generated in two files which differ only in the name of the subroutine and of course in the name of the files).

When programming, the following variables can be directly considered but they can not be changed and their names must be the same as stated below:

T0 - basic sampling time of the structure,
T - current time (T=0, T0, 2*T0,...),
K - current sample (T=K*T0),
IC - parameter for the detection of Initial Conditions); it can have the values:
  1- data reading, initialization, calculation of the output at time 0,
  0- normal simulation,
  -1- initialization, calculation of the output at time 0 (used for the optimization where the calculated optimal values of the parameters must not be redefined with a new reading),

IKONEC - parameter for the detection of the simulation run termination where it can have the values:

0 - the loop will be executed at least once more (each subroutine will be called at lest once more),

1 - last call of the subroutine.

In each subroutine all the inputs, outputs and parameter vectors of all blocks can be used. Therefore the user should be familiar with their names (to be correctly defined in the user subroutine call) and structure (to be correctly used in the subroutine itself).

For the variables (input or output of a block) the following is valid:

- The name of the variable in the program is the same as the name written in the simulation scheme after finishing the structure definition (on the screen, e.g. UU).
- All variables are organized as vectors (even if the variable is univariable it is declared with dimension 1), their dimension parameter however is named in such a way that the letter N preceds the name (e.g. NUU).

For the blocks represented by rectangles after finished definition of the simulation scheme the following is valid:

- The parameter vector of the block (for e.g. block named on the scheme PR) is denoted by letter P followed by the name of the block (e.g. PPR); the description of the dimensions and structure of parameter vector for linear blocks (TGZ, TMZ, CMA) is given in the point f of this remarks, while for user blocks (TBL) they are defined by the user himself.
- In the case of linear model TGZ, the vector denoted by the letter N followed by the name of the block (e.g. NPR) with dimension (M*L+1) contains the "orders" of the transfer functions in the already described order. For linear model TMZ this variable (e.g. NPR) is not defined as a vector and represents the order of the system.
- The state vector of the block is denoted by the letter X followed by the name of the block (e.g XPR). For linear models TMZ this vector contains the old state vector $\underline{x}(k-1)$ (see the description of the method for the transformation TMZ -> TTI) and has dimension NP. For linear models TGZ this vector contains the auxiliary variable delays (for univariable models the variables $r_1(k-n)$ $[n=1,NP_{G0}]$, for multivariable models first delays $w_{ij}(k-n)$ $[(((n=1,NP_{Gij}), i=1,L), j=1,M)]$, followed

by the delays $r_i(k-n)$ $[(n=1,NP_{G0}),i=1,L)]$; (see the description of the method for the transformation TGZ -> TTI) having the dimension NP_{G0} for univariable and dimension $L \cdot NP_{G0} + \sum_{i,j} NP_{Gij}$ for multivariable systems respectively. In the definition of the transfer functions orders (NP_{Gij} and NP_{G0}) the parameter description in pt. f of this remarks should be considered.

The described general characteristics for rectangle blocks of the simulation structure are valid for the blocks representing linear models (TGZ, TMZ, CMA) or user blocks (TBL). In general they are not valid for implemented special control elements (e.g. Kalman filter or Kalman observer).

Beside general characteristics of the newly linked program the following specifics should be taken into account:

- If the user block is written for the block process (PR) then thedelay which is for one interval of the basic sampling time (T0) less than the actual one must be realized in it (due to the construction of the program).
- In the subroutines representing the estimator or observer it must be considered that the process (PR) in the program has the inputs and the outputs shifted for one step (due to the sorting algorithm at the call of our subroutine, the output of the process in the current (actual) moment is known as well as the input to the process for the previous step (T-T0)).
- Subroutine containing the optimization constraints must correspondingly set the program parameter with the name LIIII. If the constraints are not violated LIIII=0. By the value of LIIII however, (LIIII≠0 and LIIII≠1) the constraints violation can be identified (value LIIII=1 is not allowed, it is reserved for the internal detection of instability - overflow protection). The unique identification of particular constraint violation is useful for the easier adjustment of the optimized parameters initial values.
- The user must take care of the correct execution moments for the user blocks (changes of the structure's basic sampling time may cause essential changes of the complex feedback system's behaviour if not enough attention is paid to the construction of the user blocks or to the redefinition of the basic sampling time of the structure).

h) Data base for the structure definition

The exact content and the form of the data base, which is the result of the structure definition for the first simulation and optimization run respectively (group of files STRUKT.UN*) is not important for the user. Here only the basic features will be given which enable easier identification of the errors at the repetitions of the runs.

The group of files STRUKT.UN* consists of:

STRUKT.UNI - containing the basic information about the scheme shape,
STRUKT.UN1 - containing the list of all variables and their dimensions,
STRUKT.UN2 - containing all nonstructural parameters for the simulation,
STRUKT.UN3 - containing nonstructural parameters for optimization option (except weighting matrices for the general quadratic criterion),
STRUKT.UN4 - containing weighting matrices for the general quadratic criterion ,
STRUKT.UN5 - the newly linked executable program version,
STRUKT.UN6 - data needed for the Kalman filter or Kalman observer implementation,
STRUKT.UN7 - source code of the newly linked program,
STRUKT.UN8 - data base for UNICUS compiler (the result of the UNICUS interpreter).

The files with the extensions UNI, UN1, UN2 and UN5 must exist for every structure to be repetitively run. The files *.UN3, *.UN4, *.UN6 however are needed only in the cases where the corresponding "option" or "model" is included in the structure. Files *.UN7 and *.UN8 are not important for repetitive runs.

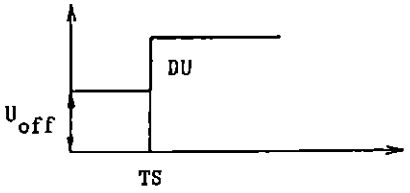
i) Standard sources

The same set of standard sources can be used at all simulations in the package (transformations of the models and simulation of the general simulation structure). For more succesfull work the standard sources are modified in comparison with their classical definition. In the sequel a brief description of all possible standard sources based on the corresponding figures of their time courses is given.

The sources which are independent of the sampling time ("continuous") are:

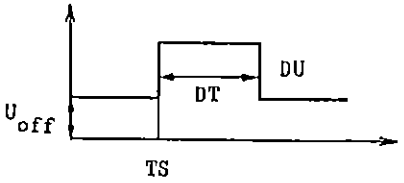
1) Step

U_{off} - offset
DU - step amplitude
TS - jump appearance time



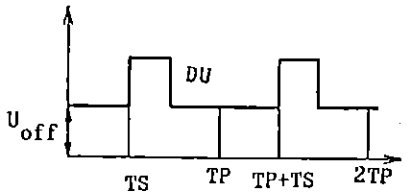
2) Pulse

U_{off} - offset
DU - pulse amplitude
TS - pulse appearance time
DT - pulse duration time



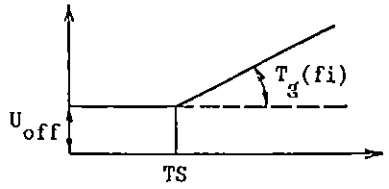
3) Train of pulses

U_{off} - offset
DU - amplitude of pulses
TS - pulse appearace time inside one period
TP - period



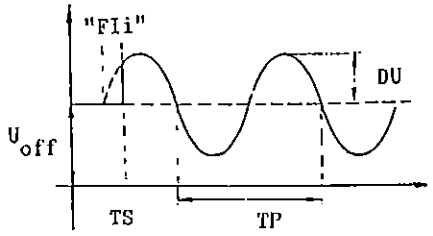
4) Ramp

U_{off} - offest
$T_g(fi)$ - slope (tan of rising or falling angle)
Ts - ramp appearance time



5) Sinusoidal (harmonic) signal

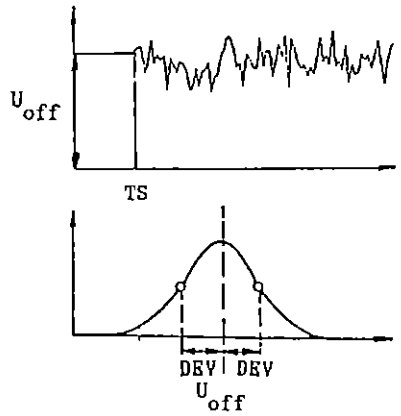
U_{off} - offset
DU - amplitude of the wave
TS - wave appearance time
FIi - phase shift (in degrees)



Sources which depend on sampling time ("discrete")

1) Gaussian noise (pseudorandom signal with normal or gaussian probability distribution)

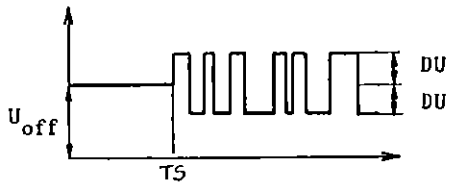
Uoff - offset
DEV - standard deviation
TS - noise appearance time
TOb - moment of noise calculation ("sampling time")
IN(1) - initial conditions for
IN(2) the random signal generator



For TOb, I1, I2 see the remarks at the end of the description of the sources.

2) Pseudorandom binary signal (PNBS)

Uoff - offset
DU - PNBS amplitude
TS - PNBS appearance time
TOb - moment of noise calculation ("sampling time")
IN(1), - initial conditions for
IN(2) 32 bit shift register



For TOb, I1, I2 see the remarks at the end of the description of the sources.

Remarks:

a) The "moment of noise calculation" (at gaussian and pseudorandom binary signal) means the time instants in which the source value is changed (between these moments, i. e. "sampling times" the source value is constant). Parameter TOb is included due to the possibility of defining the bandwidth of the noise (which can be influenced only by the "sampling time").

b) The initial conditions IN(1) and IN(2) (at gaussian and pseudorandom binary signal) determine the sequence which is generated by pseudorandom generator (with the same initial conditions this sequece is always

the same). If several uncorrelated sources are needed, different initial conditions for the generator and register respectively must be defined.

4. CONCLUSION

In conclusion we can say that the transfer of the program package ANA to the IBM-PC type of personal computers succeeded entirely and that the package in this version is already in use. It is also our opinion that the present concept of the existing package, especially the concept of the general dynamic simulation structure represents a good basis for development of a new advanced package in the frame of further German - Yugoslav cooperation.

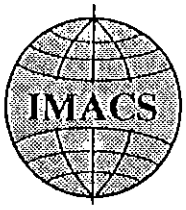
6. APPENDIX

IMACS 1988

12TH WORLD CONGRESS ON SCIENTIFIC COMPUTATION

PROCEEDINGS

JULY 18-22 1988, PARIS, FRANCE



TECHNICAL PAPERS
AUTHORS INDEX

SIMULATION ASPECTS OF INTERACTIVE PROGRAM PACKAGE ANA

M. Šega, S. Strmčnik, *M. Atanasijević, *D. Matko, *B. Zupančič, *R. Karba
J. Stefan Institute, Jamova 39, 61000 Ljubljana, Yugoslavia
*Faculty of Electrical Engineering, Tržaška 25, 61000 Ljubljana, Yugoslavia

Abstract: The aim of the work is to represent simulation capabilities of our universal interactive program package ANA for computer aided analysis and control system design. Due to ANA's concept we distinguish two "different" ways, i. e. simulation of dynamical models (transformation of the model to its input output data equivalent) and simulation of general dynamical structures. Simulation as transformation helps especially for fast interactive model verification. Simulation of general dynamical structures is rather original approach and is realized with so called UNICUS. UNICUS enables on-line and off-line simulation as well as off-line optimization. In UNICUS each dynamical subsystem of concrete structure can be defined in one of several permissible presentations. Because general models (user written FORTRAN subroutine, a block written in simulation language SIMCOS and also a real model or analog computer) are permissible, treatment of nonlinear, time varying systems is also enabled. Structure and parameter (re)definition is user friendly. The work contains descriptions of ANA's simulation capabilities where emphasis is on UNICUS. The brief description of the developed simulation library which includes also CSSL-like simulation language SIMCOS is given. At the end of the work some of ANA's capabilities are illustrated through "hybrid" simulation of controlled semibatch distillation pilot plant.

Introduction

Over the last decade a large amount of work was devoted to the development of computer-aided-design tools because it is obvious that the variety of control methods and algorithms is so wide that an engineer could not master both areas i.e. the control theory and necessary programming (1).

Our interactive program package ANA for analysis and control system design (2,3) is designed for univariable or multivariable, continuous or discrete systems. The fundamental principles for computer aided design software development were taken into account, namely the package modularity, flexibility, interactiveness and portability (4). It includes all necessary operations from control theory (transformations among various system descriptions, analysis, synthesis) and appropriate I/O facilities. All operations in the package are based on following representations (models) of dynamical systems:

a)sampled data time series of inputs and outputs,
b)linear models
 -transfer function matrix as continuous model,
 -transfer function matrix as discrete model,
 -state space description of continuous model,
 -state space description of discrete model,
c)general representations used in simulation
 -physical system model (real model or analog computer),
 -model written in digital simulation language SIMCOS (5) is called simulation block,
 -model written as a FORTRAN subroutine is called user's block.

The usefulness of the package is shown especially through the great interactiveness. In our opinion the simplest man-machine dialogues in connection with BATCH processing are appropriate for differently skilled users as well as for educational purposes. The program package is written mainly in standardized program language FORTRAN-IV so the portability among various systems is relatively easy. It is already implemented on five operational systems (RT-11, RSX-11M, VMS, Delta-M and PC-DOS).

From control system theory it is well known that simulation is very important step in verification of models, designed controllers and controlled structures. It is also unavoidable during nonlinear time varying system treatment. So ANA's simulation capabilities are very wide and they are realized in rather original way with the idea to integrate several different simulation tools. Due to ANA's concept two "different" ways of simulation are distinguished:

-off-line model simulation,
-on-line or off-line simulation of general dynamical structures is realized through so called UNICUS (6); for off-line applications also optimization is enabled.

Simulation of models

Simulation of all linear models is introduced especially for fast interactive model analysis and verification. Conceptually it belongs to the transformations because irrespective of the model type and space (continuous and discrete) the simulation result is always equivalent model represented by input output data time series.

In the package the model simulation is realized through independent program unit for each model type and it is based on the simulation library.

General dynamical structure simulation

Simulation of general dynamical structures with possibility of off-line optimization is realized through UNICUS which consists of compiler and interpreter. Because of its complex function and the fact that for instance controllers and compensators can be optimized with UNICUS we sort it in synthesis operations.

UNICUS compiler

For a new structure or structural changes in an old one UNICUS compiler generates on the basis of certain input data (stored in the file) a new source program (and subroutine for off-line optimization) in FORTRAN. The compilation is done in several steps where various tests of the simulation scheme validity and sorting procedure are performed.

The compiler is block oriented where multivariable blocks are connected in the structure by means of inputs and outputs and where a block can be either a:

a)Linear or general system model
 Continuous or discrete linear models are results of previously executed operations in ANA. General models that describe desired input output system behaviour should be prepared prior ANA entering.
b)Source of signal
 During the simulation a source of signal can be one of some standard signals (step, pulse (train), ramp, sinusoidal signal type, Gaussian or pseudorandom binary noise) or a signal generated (recorded or computed) by ANA.

c)Signal joining block (summing and subtracting)
d)Special block
State estimator or Kalman filter as the result of certain synthesis operation in ANA or parameter estimator using recursive least squares, recursive extended least squares or recursive maximum likelihood algorithm can be included.

In the case of optimization for off-line applications after each simulation run the criterion function is evaluated and used in subsequent step for optimization with installed searching method. The criterion function can be chosen from a menu or written by user in the form of a FORTRAN subroutine. As the optimization algorithm enables optimizations with arbitrary constraints, they can be introduced in UNICUS by FORTRAN logical expressions or by a user written subprogram.

UNICUS interpreter
UNICUS interpreter ensures regular running of whole UNICUS being incorporated in the package ANA. It leads the user interactively through all possible paths to minimize error probability. Namely it makes all necessary tests of the scheme and parameter validity. It also executes all operations that do not depend directly on concrete structure without user's intervention. So from users point of view it seems that the whole program package ANA including UNICUS and generated simulation programs is only one user friendly interactive program.

The main functions of UNICUS interpreter are:

a)For a new simulation scheme:
-Enables simple definition of structure.
-Enables the definition of structural parameters, what means parameters which can not be changed without repeated UNICUS compiler run (like model types and for general models their structural definitions, logical expressions as the simulation timing conditions, optimization option, constraints during optimizations, etc.).
-Enables the definition of nonstructural parameters which can be redefined without changes in the executable program version.
-On the basis of structure definition, structural parameters and nonstructural parameters it generates the data base for UNICUS compiler.
-Starts the UNICUS compiler, FORTRAN compiler, linker and finally UNICUS interpreter itself.
-At the end it starts the new simulation program which finally returns control to the supervisor of ANA.

b)For an old simulation scheme the executable version of program for simulation is already generated. So the scheme appears on the screen and redefinition of nonstructural parameters is enabled. After the redefinition it starts the simulation run. The nonstructural parameters are:
-Linear models which should be of the same type as during scheme definition.
-Initial conditions on states for state space models.
-Types and parameters of standard signal sources.
-Simulation timing conditions (initial time, final time).
-Output variable list and the communication interval.
-In the case of optimization the initial values of optimized parameters, the initial direction of searching, conditions for stopping the optimization run and for menu selected criterion functions also some associated parameters.

Simulation library

The simulation library is divided in two parts, so called executable simulation library and the SIMCOS processor.

Executable simulation library
The object code of executable simulation library is linked to the new program (and subroutine) and optionally with user written subroutines (user's block, simulation block, user written criterion function or user written subprogram that introduces constraints in optimization). In the linking process also some routines from ANA's common library are included (e.g. routines for communications with adequate input and output units) as well as the mathematical library.

SIMCOS processor
Simulation language SIMCOS (SIMulation of CONtinuous Systems) is CSSL like with some advanced features that make it closer to hybrid computer simulation. So the controlled integrators with three states (operate, initial condition, hold) were added to extend the applicability of simulation. The digital blocks (discrete PID, discrete general transfer function, delay, sample and hold) were added to extend the capabilities to the simulation and design of digital control systems. All standard nonlinear functions are also included in the language. In spite of the fact that SIMCOS can be an autonomous simulation language it was developed mainly to extend the simulation capabilities of ANA.

SIMCOS processor is used before ANA starting to transform the program written in simulation language into adequate FORTRAN subprogram which is used as a general block in UNICUS. SIMCOS processing is realized similar as UNICUS compilation, namely in several steps with syntax and scheme validity (on the basis of sorting algorithm) tests. During repetitional runs of the same program some nonstructural parameters can be also changed.

Semibatch distillation column control

Distillation is typical energy consuming process. Even for binary distillation column, the successful control of the composition of both top and bottom product can yield substantial profit resulting from the potential saving in utility cost and of course no man supervising is needed in such cases.

For the separation of components on the basis of volatility, different kinds of distillation are used in chemical industry among which semibatch distillation can in some cases (7) be very efficient.

The process under consideration consists of reboiler, distillation column with 9 plates, condenser and reflux distributor which returns one part of liquid from the condenser to the column and the other part is drawn as top product. The plant has multivariable properties which can not be neglected in the process of modelling. The governing equations for the dynamic nonlinear model can be derived by application of material balance and equilibrium equations (7). Such model has been tested by the aid of digital simulation language SIMCOS. Because of relatively slow dynamics of the process the linearisation is justified, of course only in the near neighbourhood of the chosen working points. Through the study of the plant it was established that a model of input-output behaviour of sensible less order can be obtained by black box identification method (3). The resulting system can be described in state space form, where matrices in our case are as follows:

$$A = \begin{bmatrix} -0.435 & 0.438 & 0.017 & -0.019 \\ -0.123 & 0.121 & -0.009 & 0.010 \\ -0.198 & 0.193 & -0.343 & 0.353 \\ 0.102 & -0.097 & -0.170 & 0.161 \end{bmatrix} \quad B = \begin{bmatrix} 0.126 & -0.097 \\ 0.118 & -0.080 \\ 0.292 & -0.117 \\ 0.220 & -0.093 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad (1)$$

The control problem of the plant in this case consists of maintaining two output variables, the compositions of the distillate and the bottom product, at some desired values by manipulating the reflux flow rate and the boiling rate. Very good solution can be the use of multivariable output feedback PI controller (8). For the reduced model (1) the following controller matrices were computed:

-for P-part:
$$K = \begin{bmatrix} -0.924 & 5.119 \\ -0.293 & -8.005 \end{bmatrix}$$

-for I-part:
$$Q = \begin{bmatrix} -5.885 & 8.335 \\ 0.327 & -2.086 \end{bmatrix} \qquad (2)$$

The final goal of our work is computer-controlled pilot plant, so the closed-loop system was tested also by the aid of "hybrid" simulation facilities of UNICUS and SIMCOS in ANA. Fig. 1 shows interactively defined structure for the simulation with UNICUS. Block PR is distillation column model (1) and was realized with simulation language SIMCOS. Block R2 contains

digital controller (in the form of discrete transfer functions) obtained from (2) with discretization procedure in ANA. Results of "hybrid" simulation are shown in Fig. 2, where the system was excited with initial conditions (at t = 0.).

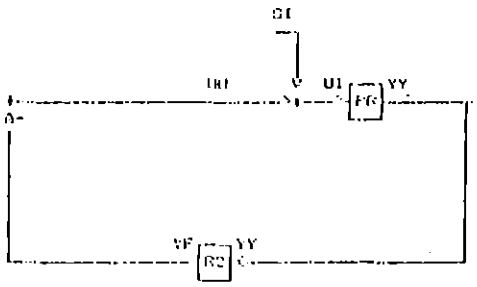


Figure 1. Simulation structure for "hybrid" simulation with UNICUS.

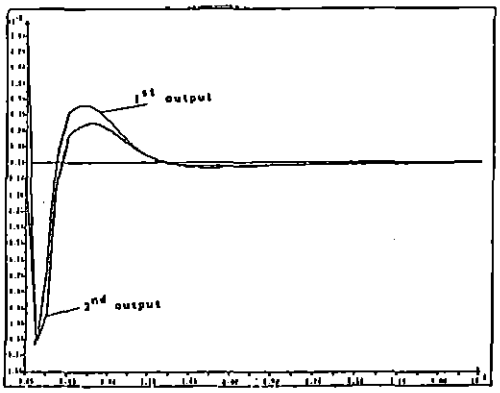


Figure 2a. Output time responses of closed loop system.

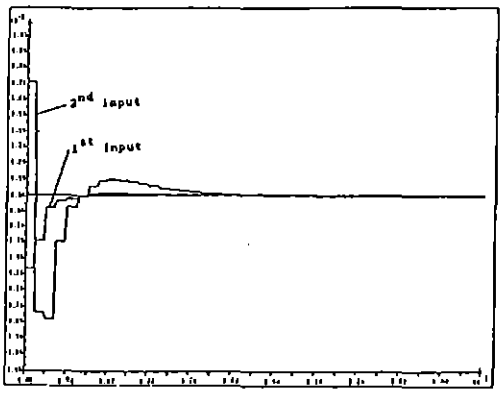


Figure 2b. Control inputs.

Conclusion

In the work simulation capabilities of the interactive program package ANA for analysis and control system design are described. In conjunction with other operations from control system theory and powerful input output facilities they are already used for research and application projects and also for educational purposes.

Besides the fact that several simulation tools are integrated in ANA the essential and rather original contribution of UNICUS is that they can be used in mixed manner (including so called general blocks: programs written in FORTRAN or in simulation language SIMCOS, real models or analog computer). So on-line and off-line simulation of complex structures is enabled. For off-line applications the optimization which is for control system design as necessary as pure simulation is also enabled.

In our opinion described general simulation scheme and parameter (re)definition in UNICUS is also an advantage especially for unexperienced users. Due to the fact that interpreter "leads" the definition in an user friendly way based on semigraphics and question and answer dialogue the probability of mistakes is minimized.

Finally we can say that powerfulness of UNICUS program complex with associated SIMCOS processor was proved on simulation of several real systems like that one shown in the example. But nevertheless both UNICUS and SIMCOS are still under development. For example in UNICUS we intend to introduce some other methods for parameter estimation and some other optimization algorithms. SIMCOS will be improved especially by introducing new integration procedures and some other control oriented operators.

References

(1) Frederick, D.K. (1982). Software summaries on computer aided control system design software packages. Control Systems Magazine, Vol. 2, No. 4, pp. 37–44.

(2) Šega, M., S. Strmčnik, R. Karba and D. Matko (1985a). Interactive program package ANA for system analysis and control design. Prep. CADCS'85 IFAC Copenhagen Symp., pp.145–150.

(3) Šega, M., S. Strmčnik, R. Karba and D. Matko (1985b). Control system treatment by program package ANA. Proc. IMACS Oslo Symp., Vol. 4, pp. 95–98.

(4) Wieslander, J. (1979). Design principles for computer aided design software. Proc. IFAC Zurich Symp., pp. 493–496.

(5) Zupančič, B., D. Matko, M. Šega and P. Tramte (1986) Simulation in the program package ANA, 2nd European Simulation Congress in Antwerp, pp. 314–318.

(6) Matko, D., M. Šega and B. Zupančič (1985). UNICUS- A new approach to the simulation and design of control systems. Proc.IMACS Oslo Symp., Vol. 4, pp. 91–94.

(7) Šega, M., M. Atanasijević, R. Karba and M.Milanović (1986). Computer aided design of semibatch distillation column control, 2nd European Simulation Congress in Antwerp, pp. 628–634.

(8) Atanasijević, M., R. Karba, F. Bremšak, T. Recelj, J. Golob and L. Fele (1986). Comparison of three different approaches to the semibatch distillation column control design, IFAC Symposium in Bournemouth, pp. 231–236.

Scientific Series of the International Bureau
KERNFORSCHUNGSANLAGE JÜLICH GMBH

Climatic Zones and Rural Housing in India
Editors: N.K. Bansal, Gernot Minke
GERMAN-INDIAN COOPERATION
ISBN 3-89336-008-5

Titanium Nitride Coatings
Preparations, Characteristics and Applications
S. Marinković, Z. Marinković and H. Kötter
GERMAN-YUGOSLAV COOPERATION
ISBN 3-89336-010-7

The Nappe Structure of the North Sporades in Greece
The Glossa Unit of Skopelos
V. Jacobshagen and D. Matarangas
GERMAN-GREEK COOPERATION
ISBN 3-89336-015-8

Impact of Green on the Urban Atmosphere in Athens
M. Horbert, A. Kirchgeorg
A. Chronopoulou-Sereli, J. Chronopoulos
GERMAN-GREEK Cooperation
ISBN 3-89336-016-6

Vertrieb: KFA Jülich GmbH, Zentralbibliothek
Postfach 1913 · D-5170 Jülich
Telefon: 02461/61-5367 · Telex: 833556-70 kfa d